USER INTERFACE SPECIFICATIONS FOR THE GLOBAL COMMAND AND CONTROL SYSTEM (GCCS)

VERSION 1.0

OCTOBER 1994

USER INTERFACE SPECIFICATIONS
FOR THE
GLOBAL COMMAND AND CONTROL SYSTEM (GCCS)

VERSION 1.0

OCTOBER 1994

Submitted by:	Reviewed by:
/s/ CINDY L. HOPKINS	/s/ TERENCE FONG
GCCS Engineer	LTC, USA
Chief Engineer, GCCS	

Approval By:

/s/ JOHN A. GAUSS
Rear Admiral, USN (DISA JIEO)
Deputy Director, Engineering
and Interoperability

USER INTERFACE SPECIFICATIONS FOR THE GLOBAL COMMAND AND CONTROL SYSTEM (GCCS)

VERSION 1.0

OCTOBER 1994

Prepared for:

Joint Interoperability and Engineering Organization
Defense Information Systems Agency

Prepared by:

Kathleen Fernandes, Ph.D.
Naval Command, Control and Ocean Surveillance Center
Research, Development, Test and Evaluation Division
Command and Intelligence Systems Division (Code 42)
San Diego, CA 92152-5000

FOREWORD

The specifications contained herein define the “look and feel” for the user interface of joint command, control, communications, computer, and intelligence (C^4I) systems built in accordance with the Common Operating Environment (COE) standards defined by the Global Command and Control System (GCCS). The intent of the specifications is to document the “look and feel” of the GCCS user environment so that all aspects of a GCCS-based system will appear and operate in a consistent and predictable manner.

The specifications were developed in accordance with the Department of Defense (DoD) Technical Architecture Framework for Information Management (TAFIM) which defines an open systems framework for implementation in DoD information management systems. The specifications comply with the TAFIM in defining a user interface based on X Window and the Motif graphical user interface. In addition, the specifications are consistent with the DoD Human Computer Interface (HCI) Style Guide, as required by the TAFIM, and with military and commercial publications on user interface design. The DoD HCI style guide calls for DoD organizations to publish specifications that reflect the consensus of the organization on the “look and feel” they intend to provide in systems within a particular domain; the current document provides these specifications for joint C^4I systems developed in accordance with the GCCS COE.

The specifications are based on the DoD HCI Style Guide and the following user interface style documents published by DoD services and organizations:

Air Force Intelligence Data Handling System (IDHS) Style Guide
Human Factors Design Guidelines for the Army Tactical Command and Control System (ATCCS) Soldier-Machine Interface
Theater Battle Management (TBM) Human Computer Interface (HCI) Specification
Department of the Air Force, Standard Systems Center (SSC) Graphical User Interface (GUI) Standards
User Interface Specifications for the Joint Maritime Command Information System (JMCIS)
Operations Directorate Graphical User Interface Standards.

The current document is the first publication of user interface specifications to be implemented by all GCCS-based systems. As such, there are a number of topics that it does not yet address. Some of these topics are identified in footnotes, while others are explained in "Note to reviewers" comments displayed in italics in the section where the topic will be addressed. Sections where some or all of the content is not yet available include:

Assumptions regarding hardware and software configuration, the GCCS concept, system integration under GCCS, and the operational environment (section 1.3),
Style guide compliance and configuration management (section 1.6),
Keyboard mappings for each hardware configuration (section 2.3.2.2),
Color sets for system and application windows (sections 7.2 and 8.3.1 and appendix A), and
System integration using a desktop model (section 7.3).

Comments concerning the specifications should be directed to Commanding Officer, Attn: Dr. Kathleen Fernandes, NCCOSC, RDT&E Division, Code 4221, 53560 Hull Street, San Diego, CA 92152-5001, (619) 553-9224 (same for DSN), fax: (619) 553-5136, e-mail: fernande@nosc.mil. Comments should be submitted in the following format:

Section/paragraph/sentence addressed:
Wording of proposed change:
Rationale for proposed change:

Individuals should include their name, organization, and phone and fax numbers when they submit their comments.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

1.0 INTRODUCTION

1.1 BACKGROUND

The U.S. Department of Defense (DoD) is developing and fielding the Global Command and Control System (GCCS) to support joint command, control, communications, computer, and intelligence (C^4I) decision making as envisioned under the C^4I for the Warrior concept. GCCS is composed of a set of application programs that can be configured to meet the needs of the joint forces commanders who use the system. Some applications provide generic functionality that can be shared among the user community, while others are targeted to the unique needs of individual users.

It is critical to the overall usability of joint C^4I systems such as GCCS that the applications being developed have a user interface with a common appearance and behavior. The user interface needs to be standardized so that the system will appear and operate in a consistent manner. In addition, users should be able to navigate easily both within and between GCCS applications as well as apply their experience with one application to other C^4I applications and systems.

The user interface provides the link between users and an application and must be designed so that users can access and manipulate data easily, efficiently, and with minimal error. Interface standardization is particularly important for C^4I systems such as GCCS as users are provided the capability to interact with a variety of complex, multi-windowed applications within a single system. The benefits to be gained from standardization are increased user productivity, reduced training requirements, improved system reliability, and increased efficiency in the development of individual applications as well as entire systems.

1.2 PURPOSE

This document defines a common appearance and behavior for the user interface of GCCS-based C^4I systems. The specifications presented in this style guide identify the types of objects users will see and the actions users will take to interact with these objects. The document focuses on the appearance and behavior of the user interface and does not address environmental specifications in areas such as lighting and noise, or ergonomic specifications in areas such as workstation design and layout.[1]

The objective of the specifications is to establish a common "look and feel" for the GCCS user interface. That is, the user interface should be designed so that when users see an object on the screen, they are able to recognize both the type of function they can perform with the object and the means to perform the function. For example, a typical function performed by users is to change the size of a window. When an interface has a common "look," users know that any window with a resize border can be resized and, conversely, any window without a resize

[1]. Although this document does not contain environmental criteria with which GCCS must comply, the specifications presented here have been developed to be appropriate for the range of environmental conditions (e.g., illumination) in which the system will be installed.

border cannot be resized. When an interface has a common “feel,” users know that whenever they see a resize border on a window, they can change the size by dragging the window border and that whenever they can resize a window, they do so by following this same method.

Implementing a common “look and feel” enables users to identify, remember, and predict the rules and organization of a system. By building consistency in the user interface, users can develop an effective and efficient model of how the system works. According to Mayhew in Principles and Guidelines in Software User Interface Design (p. 97), a consistent user interface is one that provides:

Consistent location of certain types of information on screens,
Consistent syntax of commands in a command language,
Similar execution of analogous operations in different applications,
Consistent design of command names and abbreviations,
Consistent grammatical form of error messages and instructions,
Consistent design of captions and fields on forms and displays,
Consistent dialog style for different functions, and
Terminology consistent with the users’ existing vocabulary.

1.3 ASSUMPTIONS

1.3.1 Hardware and Software Configuration

The user interface specifications presented in this document assume a hardware and software architecture consistent with the Common Operating Environment (COE) standards defined by GCCS. The primary hardware configuration is expected to consist of a workstation with at least one color monitor, a keyboard, and a pointing device (such as a mouse or trackball) with two or three buttons. The primary software configuration is assumed to be a POSIX-compliant operating system such as UNIX, X Window as the windowing system, and Motif as the graphical user interface. These specifications define a user interface “look and feel” based on Motif release 1.2, except where noted, and provide backward compatibility with Motif release 1.1.[2]

Note to reviewers: The specifications currently address Sun 4 and Hewlett Packard 700 series computers, each with one or more 19-inch color graphics monitors with 1280 x 1024-pixel resolution. Where available, specifications are provided to address unique display requirements for large-screen displays and smaller monitors such as those on a laptop or personal computer. The style guide will be modified as needed to address the range of hardware platforms supported by GCCS (including those with nonstandard interfaces, to the extent that they are supported within the GCCS COE). Similarly, while it is assumed that the primary

[2]. Unless otherwise indicated, references to the Motif Style Guide or the Motif User’s Guide pertain to version 1.2 of these documents.

user interface for GCCS will be Motif, this style guide will be expanded as needed to include design guidelines and a migration strategy for systems using other interface styles (e.g., character-based and Windows-based).

1.3.2 The GCCS Concept

Note to reviewers: This section will define GCCS as a collection of software modules that can be configured to support joint C4I decision making and will identify the primary players in the GCCS community.

1.3.3 System Integration Under GCCS

Note to reviewers: This section will provide an overview of the GCCS integration process as defined in the GCCS Integration Standard.

1.3.4 System Users

The specifications presented here assume that each GCCS-based system defines the categories of users to whom access will be provided. Each system specifies the functionality (i.e., specific applications) that will be available to each user category and control access to these applications during system login. For example, the functions available to a system administrator may be different than those available to normal users, and the functions available to normal users may, in turn, be configurable based on the specific tasks they perform.

The principal users of a system are expected to be operational personnel with basic familiarity with software usage but no knowledge of the operating system environment or command structure. Other user groups may include system administrators, security managers, and database administrators as required by the system.

These specifications assume that users have normal vision and hearing. However, the specifications define interface objects on the basis of a combination of features (not just color alone) and so are able to accommodate users with limited color vision.

1.3.5 Operational Environment

Note to reviewers: It is expected that GCCS-based systems will be installed in a range of operational settings (e.g., office-like spaces with normal ambient lighting, areas with bright sunlight and significant glare, and spaces where users have to remain dark adapted). This section will identify these environments and recommend that GCCS-based systems provide sufficient flexibility in customizing user interface features (e.g., selection of a display color set) so that the system is usable in the range of environments in which it will be installed.

1.4 SCOPE

This document was developed in accordance with version 2.0 of the DoD Technical Reference Model and Standards Profile (volume 2 of the TAFIM) and version 3.0 of the DoD Human Computer Interface Style Guide (volume 8 of the TAFIM).[3] The technical reference model defines the profile of open systems standards with which DoD information systems are required to comply. The DoD style guide provides guidelines for user interface standardization across applications developed for DoD organizations and calls for each organization to detail the specific "look and feel" for its systems in addenda to the guide. The user interface specifications presented in the current document serve as the addendum for GCCS.

The DoD style guide presents guidelines for application development within layers 0 through 5 of the National Institute of Standards and Technology user interface system reference model (see figure 1-1). The specifications presented here define a user interface style that is consistent with these guidelines, providing additional detail concerning layers 4 and 5 of the model as well as addressing layer 6 concerning application functions required by users. In addition, the specifications are consistent with Section 15 on user-computer interfaces in Military Standard (MIL-STD) 1472D and with Military Handbook (MIL-HDBK) 761A. Finally, the specifications are based on published user interface guidelines (such as those from Apple Computer) and parallel the style guides for C^{4}I systems prepared by DoD services and organizations.

[3]. Future versions of the specifications will be revised as necessary to ensure continued compliance with the TAFIM.

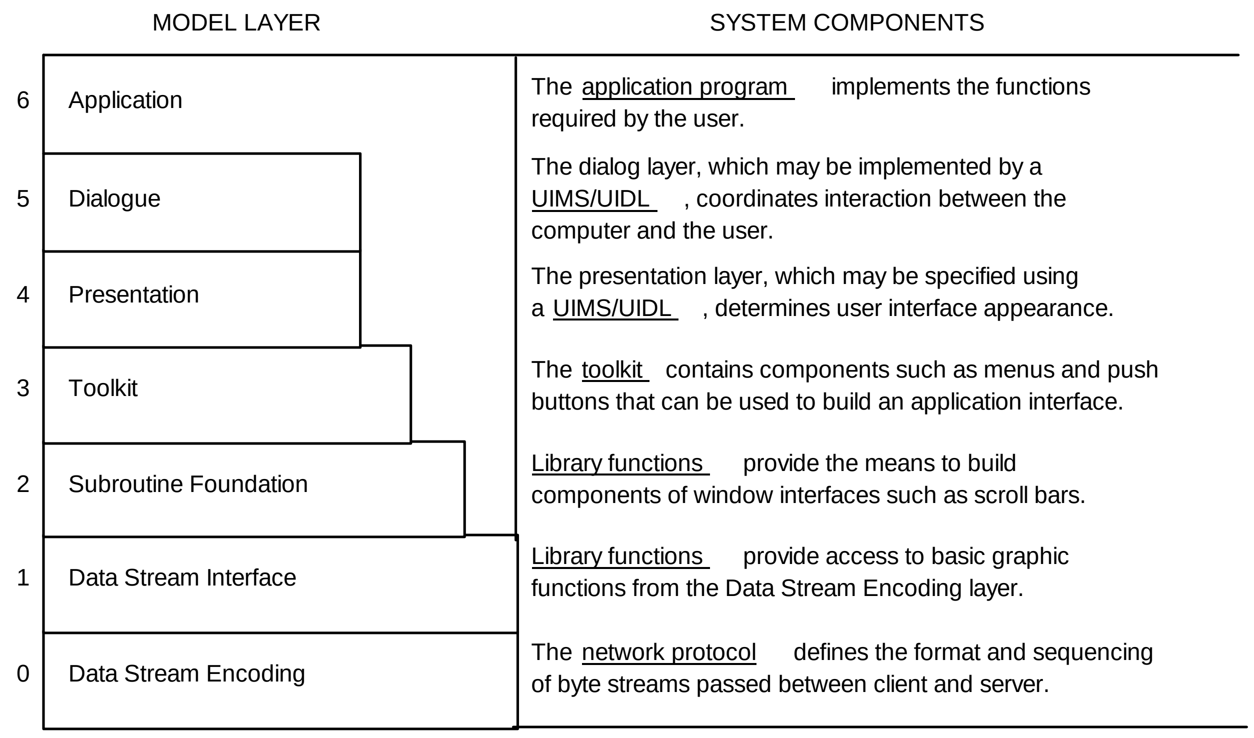

MODEL LAYER
SYSTEM COMPONENTS
6 Application
5 Dialogue
4 Presentation
3 Toolkit
2 Subroutine Foundation
1 Data Stream Interface
0 Data Stream Encoding
The application program implements the functions required by the user.
The dialog layer, which may be implemented by a UIMS/UIDL, coordinates interaction between the computer and the user.
The presentation layer, which may be specified using a UIMS/UIDL, determines user interface appearance.
The toolkit contains components such as menus and push buttons that can be used to build an application interface.
Library functions provide the means to build components of window interfaces such as scroll bars.
Library functions provide access to basic graphic functions from the Data Stream Encoding layer.
The network protocol defines the format and sequencing of byte streams passed between client and server.

Figure 1-1. National Institute of Standards and Technology user interface system reference model (from the DoD style guide).

The specifications in the current document implement a Motif-compliant “look and feel” as detailed in the Motif Style Guide, release 1.2. Deviations from the style guide were made as needed to accommodate operational requirements or constraints, provided that the deviations were consistent with established user interface guidelines.[4] When deviations are made, they are indicated in footnotes to the specifications.

1.5 INTENDED AUDIENCE

This style guide was written primarily for the user interface designers and software engineers who will create and maintain software modules for GCCS-based systems. This document is also intended for the system integrators who will design and maintain the system-level user interface for these systems. It is assumed that this audience is familiar with the basic “look and feel” of the Motif graphical user interface and has identified the intended user population and the range of tasks they will perform with an application. The specifications presented here provide a design framework for this audience to follow so that the software programs they develop provide a user interface that is consistent at both application and system levels.

1.6 COMPLIANCE

Note to reviewers: This section will address compliance requirements for GCCS -based systems, define levels of conformance with which these systems are expected to comply, and define the process for configuration management of the GCCS style guide.

1.7 DOCUMENT OVERVIEW

This document begins by specifying the manner in which users interact with an application (sections 2 and 3) and then defines the basic features of windows, menus, and controls available in the user interface (sections 4, 5, and 6). Detailed specifications are then provided on the overall design of system-level and application-level windows (sections 7 and 8) and on special design requirements for task-specific windows (section 9). Each section distinguishes those aspects of the user interface that need to be standardized and managed at the system level and those aspects that can be adapted at the application level to provide the functionality desired.

[4]. Compliance with the Institute of Electrical and Electronic Engineers (IEEE) user interface drivability recommendations will be addressed after they are formally adopted by IEEE and integrated into the DoD policy on user interface style. [5]. The Motif Style Guide requires full interchangeability between pointing device and keyboard so that all of the functionality in an application is available from both input devices. Because the current document recommends but does not require full interchangeability, the specifications are not fully Motif compliant. The DoD style guide indicates that operational military systems should provide complete interchangeability but does not define a detailed model for keyboard interaction against which to assess compliance.

Appendices include (a) the Motif resource settings for implementing the "look and feel" defined in this document, (b) a checklist of mandatory and recommended provisions so that developers can evaluate the extent to which their applications are compliant with the specifications, (c) a list of acronyms and abbreviations used in the document, and (d) a matrix of key bindings for keyboard operations defined by Motif and called out in these specifications.

1.8 TYPOGRAPHIC CONVENTIONS

The following conventions are used in the style guide:

a. The names of pointing device buttons (e.g., Select button) are capitalized as are push button names (e.g., Window Menu button) and actions (e.g., Cancel, OK) and menu titles (e.g., File) and options (e.g., Minimize, Maximize). In addition, the names of specific windows (e.g., Map Countries window) and applications (e.g., Fuel Calc application) are capitalized.

b. The names of keys on a keyboard are capitalized (e.g., Return); when the name refers to a virtual keyboard designation rather than an actual keyboard, it is shown in brackets (e.g., <Return>).

c. Simultaneous key combinations are indicated by presenting the key names separated by a plus; for example, Ctrl+Prev means that users hold down the Ctrl key and then press the Prev key.

In addition, this document uses a serifed font for both text and graphics. This font was selected for ease of readability. The graphics use the same font to be consistent with the text and should not be considered as an appropriate font for text in application software.

1.9 REFERENCES

The specifications in this document are based on the guidelines, standards, and style guides listed below.

Government Documents

Air Force Intelligence Agency. Air Force Intelligence Data Handling System (IDHS) Style Guide. Air Force Intelligence Agency, Washington, D.C., 1990.

Avery, L.W. & Bowser, S.E. (eds.) Human Factors Design Guidelines for the Army Tactical Command and Control System (ATCCS) Soldier-Machine Interface, Version 2.0. Pacific Northwest Laboratory for the U.S. Army Tactical Command and Control System Experimentation Site, Fort Lewis, WA, 1992.

Bowen, C.D. Theater Battle Management (TBM) Human Computer Interface (HCI) Specification, Version 2 (draft). The MITRE Corporation, Bedford, MA, undated.

Defense Information Systems Agency. Department of Defense Technical Architecture Framework for Information Management. Volume 2: Technical Reference Model and Standards Profile Summary, Version 2.0, 1994.

Defense Information Systems Agency. Department of Defense Technical Architecture Framework for Information Management. Volume 8: Department of Defense Human Computer Interface Style Guide, Version 3.0, 1993.

Department of the Air Force, Standard Systems Center (SSC). Graphical User Interface (GUI) Standards, Volume 1, 1993.

Department of Defense. Military Handbook 761A. Human Engineering Guidelines for Management Information System. Department of Defense, Washington, D.C., September 1989.

Department of Defense. Military Standard 1472D. Human Engineering Design Criteria for Military Systems, Equipment, and Facilities. U.S. Army Missile Command, Huntsville, AL, 1989.

Department of Defense Intelligence Information Systems Management Board. Department of Defense Intelligence Information System (DODIIS) Profile of the DoD Technical Reference Model for Information Management. Bolling Air Force Base, Washington, D.C., 1993.

Fernandes, K. User Interface Specifications for the Joint Maritime Command Information System (JMCIS), Version 2.0. Arlington, VA: Space and Naval Warfare Systems Command, 1994.

Global Command and Control System (GCCS) Integration Standard, Version 1.0 (draft), 1994.

Naval Integrated System for Exploitation (NISE) Workstation Requirements Specification, 1991.

Operations Directorate Graphical User Interface Standards. Version 1.0. Prepared by the Joint DO/DT GUI Standards Working Group, 1994.

Space and Naval Warfare Systems Command. Joint Maritime Command Information System (JMCIS) Common Operating Environment (COE), Version 1.3, 1994.

Space and Naval Warfare Systems Command. Joint Maritime Command Information System (JMCIS) Integration Standard, Version 2.0, 1994.

Non-Government Documents

Apple Computer, Inc. Human Interface Guidelines: The Apple Desktop Interface. Reading, MA: Addison-Wesley Publishing Co., 1987.

Ferguson, P.M. Motif Reference Manual for OSF/Motif Release 1.2. Sebastopol, CA: O'Reilly & Associates, Inc., 1993.

Galitz, W.O. User-Interface Screen Design. Boston, MA: QED Publishing Group, 1993.

Gardiner, M.M. & Christie, B. (eds.) Applying Cognitive Psychology to User-Interface Design. Chichester: John Wiley & Sons, 1987.

IEEE Recommended Practice for Graphical User Interface Drivability (Unapproved Draft 2), March 1992.

Kobara, S. Visual Design with OSF/Motif. Reading, MA: Addison-Wesley, Publishing Co., 1991.

Mayhew, D.J. Principles and Guidelines in Software User Interface Design. Englewood Cliffs, NJ: PTR Prentice Hall, 1992.

Open Software Foundation. OSF/Motif Style Guide. Release 1.2. Englewood Cliffs, NJ: Prentice Hall, 1992.

Open Software Foundation. Motif Style Guide, Release 2.0 (Beta draft), 20 January 1994.

Open Software Foundation. OSF/Motif User's Guide. Release 1.2. Englewood Cliffs, NJ: Prentice Hall, 1992.

Root, R.W. & McFarland, A.D. Graphical User Interface Design Guidelines for Bellcore Software Products. Issue 1. Bellcore/Bell Communications Research, Piscataway, NJ, 1993.

Smith, S.L. & Mosier, J.N. Guidelines for Designing User Interface Software (ESD0TR086-278). USAF Electronic Systems Center, Hanscom AFB, MA, 1986.

User Interface Design With OSF/Motif. Open Software Foundation Training Course, Version 1.2, 1992.

The specifications presented here make reference to the following additional documents:

Defense Intelligence Agency. Standard Military Graphics Symbols Manual (DIAM 65-XX) (Draft). Defense Intelligence Agency, 1990.

Department of the Army. Army Field Manual FM 101-5-1. Operational Terms and Symbols. U.S. Army Combined Arms Center, Fort Leavenworth, KS, 1985.

Department of the Army. AR310-50. Authorized Abbreviations and Brevity Codes. Army UPDATE Publications, Washington, D.C., 1985.

Department of Defense. Military Standard 12D. Abbreviations for Use on Drawings, Specification Standards, and in Technical Documents, 1981.

Department of Defense. Military Standard 411E. Air Crew Station Alerting Systems, 1991.

Department of Defense. Military Standard 783D. Legends for Use in Air Crew Stations and on Airborne Equipment, 1984.

Department of Defense. Military Standard 2525. Common Warfighting Symbology, Version 1, September 1994.

North Atlantic Treaty Organization Standardization Agreement 2019. Military Symbols for Land-Based Systems, 1990.

NATO Standardization Agreement 4420. Display Symbology and Colours for NATO Maritime Units, 1990.

Standards relating to the design of workstations, associated furniture, and the facilities in which they are placed can be found in MIL-STD-1472D and the following document:

American National Standards Institute. National Standard for Human Factors Engineering of Visual Display Terminal Workstations. Santa Monica, CA: The Human Factors Society, Inc., 1988.

2.0 INPUT DEVICES

2.1 INTERCHANGEABILITY BETWEEN INPUT DEVICES

Users shall be able to control computer interaction with a pointing device or from the keyboard. However, full interchangeability between input devices is not required.[5] It is assumed that users will select the input device(s) most appropriate to the task being performed. Users are expected to rely on direct manipulation, using a pointing device (such as a trackball or mouse) as the primary means of interaction for performing most operations involving object selection and manipulation. The keyboard is expected to be used principally for text entry. The keyboard shall also be available for navigation and selection, but primarily as a back-up so that users can continue to operate a system in degraded mode if the pointing device should fail.

2.2 POINTING DEVICE INPUT

2.2.1 The Pointer

The pointing device shall be used to move a pointer on the screen, select objects on which the pointer is placed, and manipulate the objects directly. The pointing device shall be associated with a single pointer which provides a graphical representation of the location of the pointing device on the screen. The pointer may change shape depending on where it is located on the screen. The hotspot of the pointer (i.e., the active point) shall indicate the precise location where pointing device operations occur (e.g., the object that will be selected when users execute a select action). Users shall control the movement of the pointer by moving the pointing device; users shall be able to move the pointer anywhere on the screen. When users move the pointing device, the pointer shall move in the corresponding direction on the screen. For most user interactions, the pointing device-to-pointer movement ratio should be close to 1:1. Users should be able to select or adjust the rate sensitivity of the pointing device as a user preference setting.

The position of the pointer should remain where it is placed on the screen until users move it; the pointer should not be moved arbitrarily by an application. If more than one screen is available, the pointer shall move from one screen to another when users move the pointing device. However, the pointer shall not move beyond the outer screen boundaries or disappear from sight. The location of the hotspot shall not move on the screen as the pointer changes from one shape to another.

2.2.2 Pointer Shapes

The shape of the pointer depends on the functionality of the area on which the pointer is located. The pointer shapes listed in table 2-1 shall be used whenever an application provides the functionality indicated in the table.[6] Applications shall redefine the pointer shape only when the pointer is within an application window.[7] The upper-left-pointing arrow shall be the general-purpose pointer for object selection in most windows. The X pointer shape, which is the default when running X Window, shall not be used by an application since displaying this pointer means that users have access to all X (and hence operating system) functions. In the remainder of this document, the pointer is assumed to be an upper-left-pointing arrow, unless otherwise indicated.

[6]. The pointer shapes in the table are a subset of the ones defined in the Motif Style Guide and are selected to provide the functionality that application developers are expected to need. Developers can use any of the Motif pointer shapes except the X pointer.

[7]. The management of the pointer shape in a multitasking environment (e.g., what application controls the pointer shape if multiple applications are running) will be addressed in a future version of this document.

Table 2-1. Pointer shapes.

Pointing

The upper-left pointing arrow is used in most window areas for object selection. The hotspot for the arrow pointer is the point of the arrow.

The I-beam pointer is used in text areas to position the text cursor and perform actions on text. The I-beam pointer is hidden during the time between any keyboard action and pointer movement (i.e., when text entry is occurring at the location of the text cursor). The hotspot for the I-beam pointer is on the vertical bar of the I-beam about one-third from the top.

Working/Caution

The watch pointer is used to indicate that an operation is being performed in a window area. When the hourglass pointer is displayed, all pointing device and keyboard actions are ignored in the area.

The caution pointer is used to indicate that action is expected in another window area before input can be made in the current area and that the pointer has no effect in the area. When the caution pointer is displayed, all pointing device and keyboard actions are ignored in the area.

Resizing

The resize pointers are used to indicate positions for area resize, with the direction of the arrow in the pointer indicating the direction of increasing size. The horizontal and vertical resize pointers indicate resize in either the horizontal or vertical direction. The diagonal resize pointers indicate resize in both the horizontal and vertical directions simultaneously. The hotspot for the resize pointers is on the elbow or the line at the position pointed to by the arrow. A resize pointer appears when the pointer is on the frame border.

Moving

The four-directional arrow pointer indicates a move operation in progress or a resize operation before the resize direction has been determined. During a resize operation, the four-directional arrow pointer indicates a direction for resizing and changes to the appropriate resize arrow when the pointer is on the frame border. The hotspot for the four-directional arrow pointer is the spot where the arrows intersect.

Sighting

The sighting pointer is used to make fine position selections (e.g., to select a location on a map display). The hotspot for the sighting pointer is the spot where the lines intersect.

Application developers shall not create a new pointer shape for functionality that already has a pointer associated with it, or use an existing shape to symbolize functionality it was not designed to represent. If necessary, developers can create a new pointer shape for functionality not listed in the table. The new shape should be easy to see (e.g., have high contrast with the background, not obscure other information on the screen), with a hotspot that is obvious and easy to locate. In addition, the new shape should provide a hint to its purpose and should not be easily confused with other objects on the screen.

2.2.3 Pointing Device Buttons

The buttons on the pointing device can be used to manipulate objects on the screen. Each button shall have a specific function that is executed whenever users press the button. The Select button shall be used to select an object, the Transfer button to transfer objects, and the Menu button to display a pop-up menu. These functions shall be bound to the left, middle, and right buttons, respectively, on a three-button pointing device. If desired, developers can use the Transfer button on the pointing device for application-specific functions (e.g., in a map window) provided that these functions do not conflict with those already bound to this button in the specifications presented here. If a two-button pointing device is used, the left button should be bound to the Select function, the right button to the Transfer function, and the two buttons together to the Menu function.[8]

Each system should provide a user preference setting so that, if desired, users who are left-handed can change the function binding to select objects with the right button and display a pop-up menu with the left button. In addition, each system should define a default double-click speed that can be easily executed by most users (so that they do not have to try repeatedly to close a window by double clicking on the Window Menu button) and allow users to modify the default as a user preference setting. Finally, the pointing device may provide a velocity sensitive gain adjustment that allows users to modify the distance the pointer moves on the screen as they move the pointing device.

Users shall perform the following actions with the pointing device buttons:

Press	Depress and hold down a button.
Release	Release a button after it has been pressed.
Click	Press and release a button without moving the pointing device.
Double click	Press and release a button twice in rapid succession without moving the poin
Move	Move the pointing device without pressing any buttons.
Drag	Move the pointing device while pressing a button.

2.3 KEYBOARD INPUT

2.3.1 The Location Cursor

[8]. These button bindings for a two-button pointing device are one of the alternatives recommended in the Motif Style Guide.

The object that has keyboard focus in a window shall be identified by a location cursor. Users shall control the movement of the location cursor from the keyboard; movement of the pointing device shall be mapped to the pointer and shall not affect the position of the location cursor on the screen. However, users shall be able to assign keyboard focus to an object with the pointing device. When users click on an object, it shall receive focus and the location cursor shall move to the object.

When a window is first displayed, the location cursor shall be placed on the control (e.g., text field or default push button) that users are most likely to select. When a window regains input focus, the location cursor shall be placed on the control that last had focus provided that the control remains available for selection; otherwise, the location cursor shall be placed on the control users are most likely to select in the window.

The shape of the location cursor depends on the type of object with keyboard focus. The box cursor, shown in figure 2-1, is the default location cursor. This cursor appears as a box around controls such as radio buttons, check buttons, push buttons, and text fields. The shadow cursor,[9] shown in figure 2-2, is used in objects, such as menus, whose outline is not normally shown. The item cursor, shown in figure 2-3, is used in controls such as lists that manage groups of elements. This cursor highlights one or more elements in the control. The fill cursor, shown in figure 2-4, is used for very small objects such as sashes in paned windows.

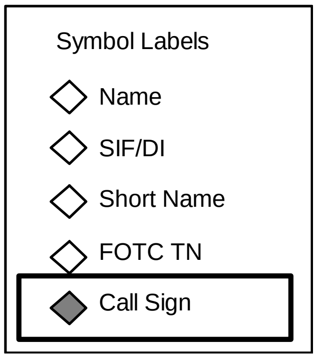


Figure 2-1. Example box cursor (from the JMCIS style guide).

[9]. The Motif Style Guide refers to this cursor shape as the outline highlight location cursor. The specifications presented here refer to this shape as the shadow cursor to minimize confusion with the highlight which is the change in color (e.g., reverse video) that occurs when an object is selected.

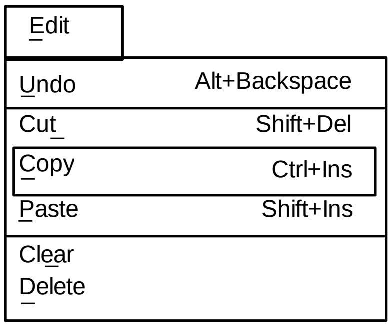


Figure 2-2. Example shadow cursor (from the JMCIS style guide).

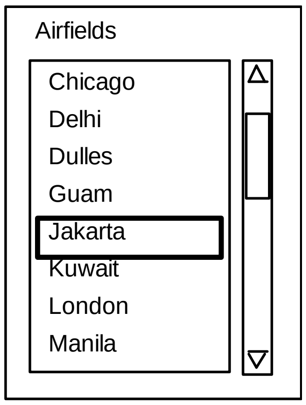


Figure 2-3. Example item cursor (from the JMCIS style guide).

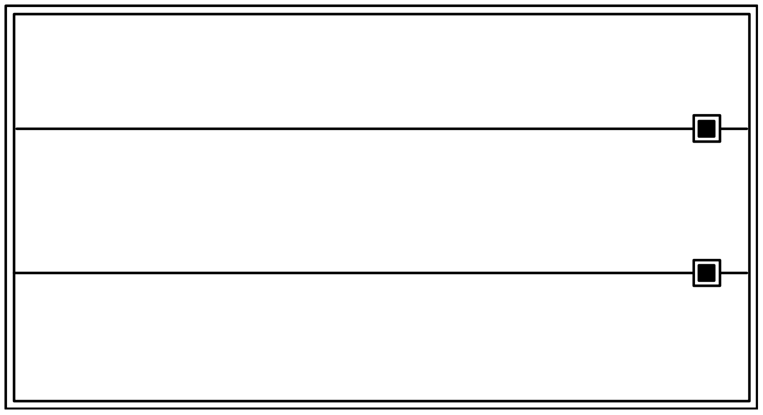


Figure 2-4. Example fill cursor (from the JMCIS style guide).

The text cursor is displayed in the text object (e.g., text field) with keyboard focus and indicates where text typed by users will be displayed. The standard shape of the text cursor shall be a vertical bar (|) in both insert and replace modes.[10] The text cursor shall flash when the object containing the text cursor has keyboard focus. The flash rate for the text cursor shall be 2-5 Hz.[11] When the text object containing the text cursor loses keyboard focus, the text cursor shall be grayed out and stop flashing. When the text object regains focus, the text cursor shall return to its normal appearance and resume flashing. In addition, if the text cursor is removed from view when a window loses focus, the cursor shall re-appear at the same location when the window regains focus.

Only one location cursor shall appear in a window at any time. Likewise, only one text cursor shall appear in a window at any time; placing the location cursor on an object that can accept text entry shall also place the text cursor in that object.

As with pointer shapes, application developers should use existing cursor shapes whenever possible to indicate keyboard focus for any new objects for which a location cursor is not already available. However, developers can create new cursor shapes as necessary to support the functionality provided by the application.

2.3.2 Key Assignments

2.3.2.1 Requirements for Degraded Mode Operation

This document uses virtual keyboard designations to indicate the key bindings for each operation executed from the keyboard. Table 2-2 lists the default bindings for the operations required by Motif for navigation and selection from the keyboard. Appendix D presents the same default bindings in matrix form and includes the keyboard accelerators shown in table 5-1.

The specifications presented here require all of the operations indicated in table 2-2 except range selection in text to be available from the keyboard in order to provide a degraded-mode operational capability. Keyboard operations to be supported in an application include navigation between and within window families, navigation and selection in windows and menus, text entry, and object transfer. The use of mnemonics and keyboard accelerators is addressed in sections 5.5 and 5.6. Range selection in text is considered optional and should be included only if this type of selection is integral to the functionality of an application and should be available to users even when they are operating a system in degraded mode.

[10]. Modification in the appearance of the text cursor (e.g., color, blink rate) to indicate the text mode currently in effect will be addressed in a future version of this document.

[11]. Motif defines the default flash rate of the text cursor, which can be adjusted through the XmNblinkRate resource, to be 500 milliseconds or 2 Hz.

Table 2-2. Motif default bindings for keyboard operations.

Virtual Keys	Operation
Window Navigation	
<Alt><Tab>	Navigates to the next window family.
<Alt><Shift><Tab>	Navigates to the previous window family.
<Alt><F6>	Navigates to the next window.
<Alt><Shift><F6>	Navigates to the previous window.
Navigation and Selection in Menus	
<F10> <Shift><Menu>	Activates/Exits a menu bar.
<Shift><F10> <Menu>	Activates/Exits a pop-up menu.
<Alt><Space> <Shift><Escape>	Activates a Window Menu.
<Space> <Select>	Displays an option menu.
<Up>	Navigates up within a menu.
<Down>	Navigates down within a menu.
<Left>	Navigates left within a menu bar or menu.
<Right>	Navigates right within a menu bar or menu.
<Return> <Enter> <Space> <Select>	Selects a menu title or menu option.
Navigation Within Windows	
<Ctrl><Tab>	Navigates to the next tab group.
<Tab>	Navigates to the next tab group except in multi-line text.
<Ctrl><Shift><Tab>	Navigates to the previous tab group.
<Shift><Tab>	Navigates to the previous tab group except in multi-line text.
<Up>	Navigates up one increment (e.g., one line).
<Down>	Navigates down one increment.
<Left>	Navigates left one increment.
<Right>	Navigates right one increment.
<PageUp>	Navigates up one page.
<PageDown>	Navigates down one page.

<PageLeft> <Ctrl><PageUp>	Navigates left one page.

Table 2-2. Motif default bindings for keyboard operations (continued).

Virtual Keys	Operation
<PageRight> <Ctrl><PageDown>	Navigates right one page.
<Ctrl><Up>	Navigates up one large increment (e.g., one paragraph).
<Ctrl><Down>	Navigates down one large increment.
<Ctrl><Left>	Navigates left one large increment (e.g., one word).
<Ctrl><Right>	Navigates right one large increment.
<Home>	Navigates to the leftmost element (e.g., the beginning of a line).
<End>	Navigates to the rightmost element (e.g., the end of a line).
<Ctrl><Home>	Navigates to the top leftmost element (e.g., the beginning of data).
<Ctrl><End>	Navigates to the bottom rightmost element (e.g., the end of data).
Selection	
<Shift><F8>	Toggles between normal and add mode.
<Ctrl><Shift><Space>	Extends a selection to the cursor position.
<Shift><Space> <Shift><Select>	Extends a selection to the cursor position except in text.
<Space> <Select>	Selects an element in a collection except in text.
<Ctrl><Space>	Selects an element in a text collection.
<Ctrl></>	Selects an entire collection.
<Ctrl><\>	Deselects all selections in a collection.
Other Actions	
<Enter> <Ctrl><Return>	Invokes a default action.
<Return>	Invokes a default action except in multi-line text.
<Cancel> <Escape>	Stops or cancels the current interaction.
<Help> <Shift><Help> <Shift><F1>	Displays context-sensitive help (if available).
Text Entry	
<Delete>	Deletes one character to the right.
<Backspace>	Deletes one character to the left.

<Space>	Inserts a space.
<Return>	In multi-line text, inserts a new line.

Table 2-2. Motif default bindings for keyboard operations (continued).

Virtual Keys	Operation
<Tab>	In multi-line text, inserts a tab or moves to the next tab stop.
<Insert>	Toggles between replace and insert mode.
Object Transfer	
<Cut> <Shift><Delete>	Cuts the current selection to the clipboard.
<Copy> <Ctrl><Insert>	Copies the current selection to the clipboard.
<Paste> <Shift><Insert>	Pastes the clipboard contents.
<Undo> <Alt><Backspace>	Reverses the most recently performed action.
<Alt><Copy> <Alt><Ctrl><Insert>	Performs a primary copy.
<Alt><Cut> <Alt><Shift><Delete>	Performs a primary move.
Range Selection in Text	
<Shift><PageUp>	Extends the selection up one page.
<Shift><PageDown>	Extends the selection down one page.
<Shift><PageLeft> <Ctrl><Shift><PageUp>	Extends the selection left one page.
<Shift><PageRight> <Ctrl><Shift><PageDown>	Extends the selection right one page.
<Ctrl><Shift><Up>	Extends the selection up one paragraph.
<Ctrl><Shift><Down>	Extends the selection down one paragraph.
<Ctrl><Shift><Left>	Extends the selection left one word.
<Ctrl><Shift><Right>	Extends the selection right one word.
<Shift><Up>	Extends the selection up one line.
<Shift><Down>	Extends the selection down one line.
<Shift><Left>	Extends the selection left one character.
<Shift><Right>	Extends the selection right one character.
<Shift><Home>	Extends the selection to the beginning of a line.
<Shift><End>	Extends the selection to the end of a line.
<Ctrl><Shift><Home>	Extends the selection to the beginning of the text.
<Ctrl><Shift><End>	Extends the selection to the end of the text.

2.3.2.2 Mapping Virtual to Actual Keys

These specifications provide keyboard assignments for the Sun 4 series and Hewlett Packard 700 series computers.[12] The keyboards for these computers are shown in figures 2-5 and 2-6. Table 2-3 maps each of these keyboards to the keys used in the Motif default bindings shown in table 2-2 and in appendix D.

Note to reviewers: This section will be modified to address the hardware configuration defined by the GCCS COE.

[12]. In X Window, the Alt key is called Meta; however, no keyboard uses a key labeled Meta. Meta is labeled Alt on 386-based personal computer, Silicon Graphics Iris, and DG Aviion keyboards, Extendchar on Hewlett Packard keyboards, and is the key with the diamond shape (next to Alt) on the Sun 4 keyboard.

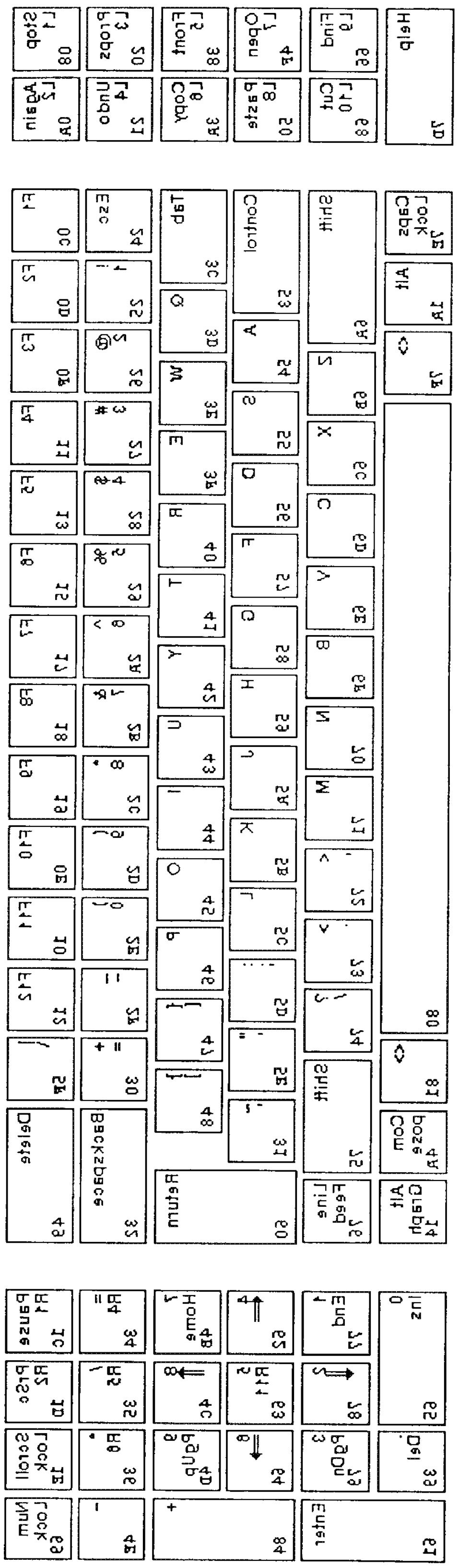


Figure 2-5. Sun 4 series keyboard.

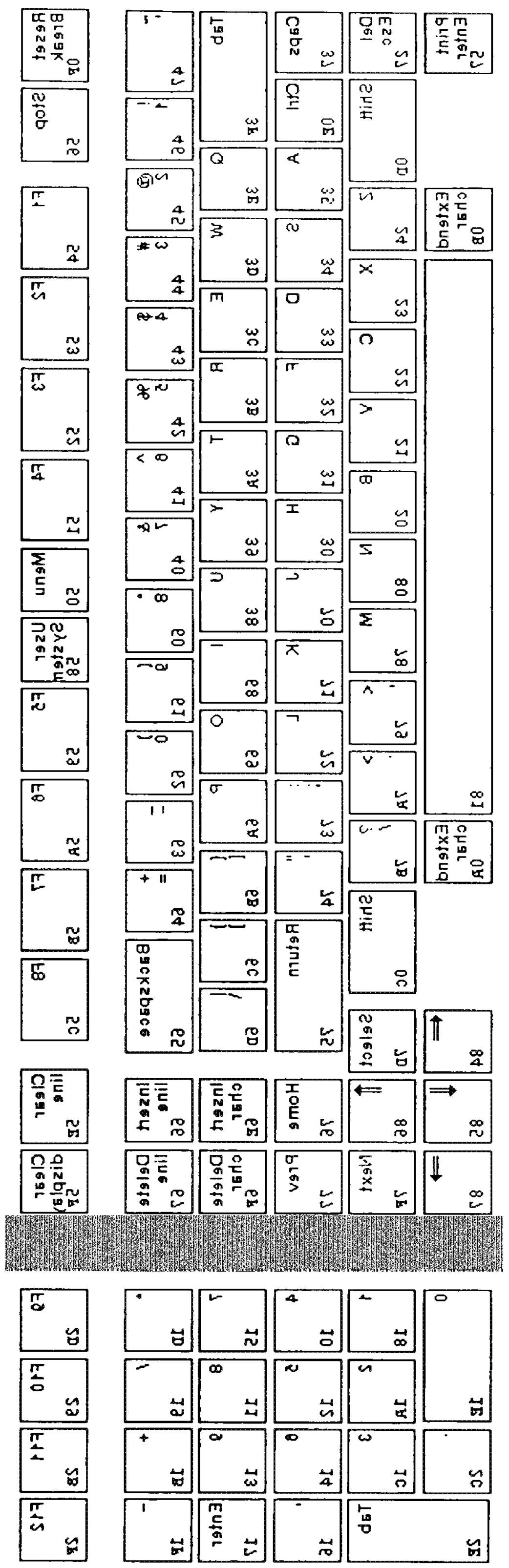


Figure 2-6. Hewlett Packard 700 series keyboard.

Table 2-3. Mapping of Sun and Hewlett Packard keys to Motif default bindings.

Virtual Key	Sun Key(s)	Hewlett Packard Key(s)
</> /	/	
<\> \	\	
<Alt>	<> (diamond-shape)	Extendchar
<Backspace>	Backspace	Backspace
<Cancel>	Esc	DelEsc
<Copy>	Copy	---- (use Ctrl+Insertchar)
<Ctrl>	Control	Ctrl
<Cut>	Cut	---- (use Shift+Delchar)
<Delete>	Delete	Delchar
<Down>	Down	Down
<End>	End	---- (use Extendchar+Right)
<Enter>	Enter	Enter
<Escape>	Esc	DelEsc
<F1> through <F10>	F1 through F10	F1 through F10
<Help>	Help	---- (use Shift+F1)
<Home>	Home	Home
<Insert>	Ins	Insertchar
<Left>	Left	Left
<Menu>	---- (use Shift+F10)	Menu
<PageDown>	PgDn	Next
<PageLeft>	---- (use Control+PgUp)	---- (use Ctrl+Prev)
<PageRight>	---- (use Control+PgDn)	---- (use Ctrl+Next)
<PageUp>	PgUp	Prev
<Paste>	Paste	---- (use Shift+Insertline)
<Return>	Return	Return
<Right>	Right	Right
<Select>	---- (use Space)	Select
<Shift>	Shift	Shift
<Space>	Space	Space
<Tab>	Tab	Tab
<Undo>	Undo	---- (use Extendchar+Backspace)
<Up>	Up	Up

Application developers shall use the mappings in table 2-3 whenever users perform any of the operations included in table 2-2.[13] If necessary, developers can create new bindings for operations not listed in table 2-2. Appendix D indicates the key combinations currently bound to keyboard operations and not available to developers. In addition, a future version of this document will address implementation of system level macros that allow users to define a sequence of commands (that cut across applications) and then map their execution to a specific key combination. This style guide reserves the combination of <Alt>, <Shift>, and an alphanumeric key for activation of these macros. Because these key combinations will be executed at a system level, developers shall not use them when defining key mappings or keyboard accelerators for their applications.

New bindings should be selected in coordination with the system(s) in which the application will be integrated to ensure consistency in these assignments across all of the applications in the system. In defining new bindings, developers should restrict the combination of <Alt> and alphanumeric characters to mnemonics and should use <Shift>, <Alt>, and <Ctrl> in combination with other keys to define keyboard accelerators. In addition, whenever possible, developers should ensure that new bindings are “visible” in a window (e.g., as mnemonics and keyboard accelerators) so that users do not have to rely on memory, use on-line help, or refer to system documentation in order to interact with an application from the keyboard.

2.3.2.3 Variable Function Keys

This style guide assumes that most applications will use fixed function keys (i.e., each key has only one predefined function associated with it) to execute the operations shown in table 2-2 and to define mnemonics and keyboard accelerators as indicated in sections 5.5 and 5.6 and shown in table 5-1. However, developers desiring to do so may also define variable function keys for providing quick access to the operations available within the application. If variable function keys are used, the command names for the function keys should be displayed in an application window in the form of soft keys, as shown in Figure 2-7. Soft keys can take on different meanings depending on the current state of the application; when the meaning of a key changes, the key labels displayed in the window should be modified to reflect the action that will be executed if the key is used. The DoD style guide recommends that if an application changes the functions assigned to a set of soft keys, it should limit the functions to no more than two per key and should include an easy means for users to return to the set of base-level functions.

[13]. The Motif Style Guide uses <F1>, <Help>, <Shift><F1>, and <Shift><Help> to provide keyboard access to help. Because these specifications reserve F1 as the keyboard accelerator for Zoom, developers should not use this key for help.

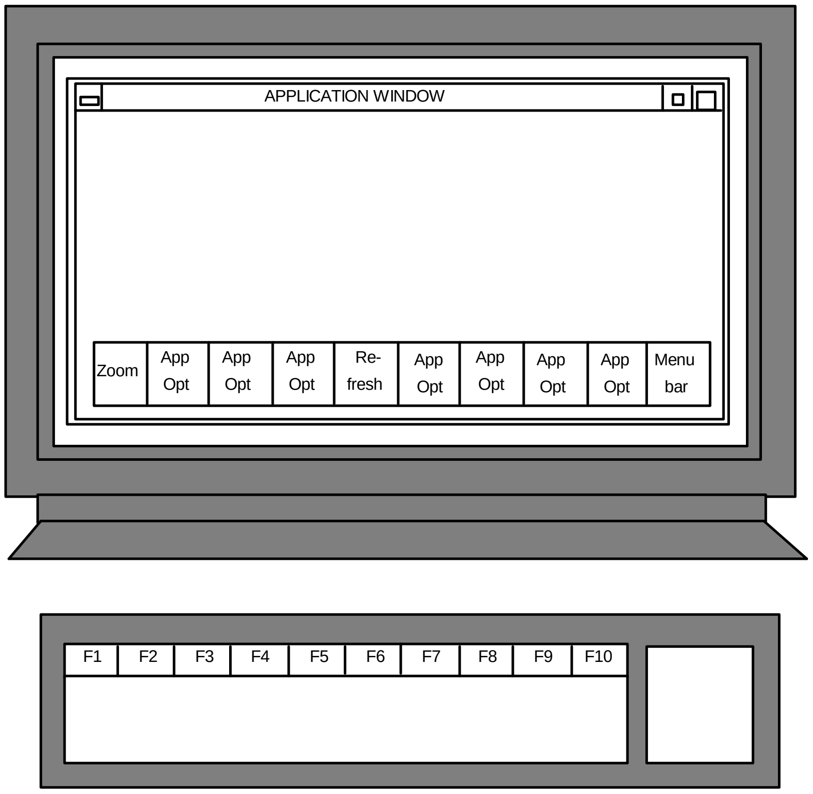


Figure 2-7. Example of soft function keys (from the DoD style guide).

If variable function keys are used in an application, they shall be implemented in a manner consistent with Motif input focus policy (see section 3.1.1) and the key mappings in table 2-2. Because the use of variable function keys is application-specific, developers need to define their implementation in coordination with the system(s) in which the application will be available and ensure that the keys do not conflict with mappings defined by other applications within the system. In addition, developers should ensure that the keys can be executed only when the application window in which they are displayed has input focus and that the keys are grayed out (to indicate their unavailability) when the window does not have focus. Developers should consider that users are likely to be working in multiple applications simultaneously, and the advantages associated with using variable function keys in an application may no longer apply when the application is one of many available to users.

2.3.3 Text Entry

2.3.3.1 The Text Cursor During Text Entry

When users move the pointer into an area where text entry is possible, it should change to an I-beam shape. To assign keyboard focus to the area, users shall click the Select button on the pointing device. The location cursor shall move to the area, and the text cursor shall be displayed in the area. If the area is empty, the text cursor shall appear at the beginning (i.e., top leftmost part) of the area. If the area contains text, the text cursor shall appear between the characters that are under the pointer. If the pointer is positioned beyond the end of the text, the text cursor shall appear following the final text character.

When a window is displayed, the text cursor shall appear in the text area (e.g., text field) where typing is most likely to occur, at the beginning of whatever text appears in the area. When users move the text cursor between text fields in the window, the cursor shall appear at the beginning of whatever text appears in each field. <Return> should not be used to move the text cursor between fields since this key is mapped to executing the default push button action in a window.

Applications should be designed so that the pointer changes to an I-beam shape only when users move the pointer into an area where text entry is possible. In addition, applications should be designed so that the text cursor can be placed only in these areas, and cannot be placed in areas where text entry is not possible (e.g. noneditable text fields). Text entry should be possible only after the text cursor is visible at a location that can accept text entry. Text entry should not be possible (i.e., should not be accepted by the application) when the text cursor is not visible. Finally, applications should ensure that the text cursor is highly visible whenever it appears in a text entry area. Visibility can be improved by changes in bolding, contrast, and size of the text cursor.

2.3.3.2 Text Entry Modes

Motif supports two modes for text entry. In insert mode, when users begin to type, the new text shall be added to the left of the text cursor, and the cursor and any existing text shall move to the right. In replace mode, when users begin to type, the character to the right of the text cursor shall be replaced with the new character that was entered.[14] <Backspace> shall delete the character to the left of the text cursor (backward deletion), and <Delete> shall delete the character to the right of the text cursor (forward deletion). Double clicking on text should select (and highlight) the word at the location of the pointer. To delete highlighted text, users press <Delete>; the text should be deleted, the text cursor should appear at the deletion point, and the remaining text should be compressed to fill in the space. <Insert> shall toggle between text entry modes. Motif uses the same text cursor shape in both insert and replace mode so there are no visual cues available to users as to the entry mode currently in effect.

In Motif, whenever the text cursor moves into a text field, the default for text entry is insert mode; users must press <Insert> to switch to replace mode.[15] Applications should

[14]. These specifications describe text entry in replace mode as it is available in Motif 1.2. Appendix A describes how to access replace mode in this version of Motif. Developers working with earlier versions of Motif should implement only the functionality available in the version they are using, rather than attempt to provide the full range of functionality described here.

[15]. A future version of this document will address how to manage the <Insert> toggle so that it can be set at other than a field level.[16]. In multi-line text, <Tab> is used for

provide access to both text entry modes so that users can select the mode that is more efficient given the nature of the text entry task being performed. For example, users should be able to select replace mode for text entry in fields with predefined attributes (e.g., latitude/longitude and date-time group), but insert mode for free text input (e.g., the text of a message). Applications should not restrict users to a single mode within a text field or arbitrarily switch between modes as users move from one field to another.

2.4 ALTERNATE INPUT DEVICES

Developers considering the use of a hardware configuration that includes an alternate input device other than a mouse or trackball should submit their requests to the appropriate configuration management board for approval prior to implementation. If the use of an alternate input device is approved, the manner in which users interact with the device (e.g., for navigation and selection) should be consistent with the interaction models presented in section 3 of this document.

3.0 USER-COMPUTER INTERACTION

3.1 WINDOW NAVIGATION

3.1.1 Input Focus Policy

According to the keyboard focus model implemented by Motif, the window that is able to receive input from the keyboard has input focus. Motif defines two modes, explicit and implicit, for assigning input focus to a window. In explicit mode, input focus can be moved among windows either with the pointing device or from the keyboard, and the keyboard can be used for navigation among the components in the window with focus. In implicit mode, keyboard focus moves with the pointer and cannot be controlled from the keyboard. Regardless of the focus policy in effect, only one window on the screen shall have input focus at any time.

These specifications call for the default input focus policy to be explicit. This focus policy is preferred in systems where several applications are running simultaneously, multiple windows are open and displayed on the screen, and users need to interact with window components both with the pointing device (e.g., point and click) and from the keyboard (e.g., typing in text fields). Although explicit focus is the default, these specifications allow implicit focus to be implemented in systems where only a few windows are open on the screen and users must transition frequently among them. In general, explicit focus should be implemented when window placement is overlapping while implicit focus is preferred when placement is tiled (see section 4.5). An application may be integrated within a range of systems and so should be able to accommodate either focus policy (e.g., not implement a selection method that requires implicit focus policy).

Given that explicit focus is required as the default, the remainder of this document describes the behavior of the location cursor under this input focus policy. Because location cursor movement is mapped to the pointer when input focus is implicit, the Motif Style Guide does not require the location cursor to be displayed when implicit focus is selected. However, because the specifications presented here provide users with the flexibility to select either focus policy, systems should display the location cursor when input focus is implicit in order to provide a consistent visual cue concerning keyboard focus regardless of the focus policy in effect.

3.1.2 Assigning Focus with the Pointing Device

Users shall assign input focus by moving the pointer into a window and clicking the Select button on the pointing device. If users click in an empty part of the window, the window frame shall highlight. If users click in the title bar of the window, the window frame shall highlight and the window shall be raised to the front of the screen. If users click on a selectable object in the window, the window frame shall highlight, the window shall be raised to the front of the screen, and the object shall be selected. A parent window shall automatically regain focus when a child window is closed; users should not have to click on the parent to assign focus to it.

3.1.3 Assigning Focus with the Keyboard

<Alt><Tab> and <Alt><Shift><Tab> shall move input focus forward and backward through the window families (i.e., primary windows and window icons) displayed on the screen, and <Alt><F6> and <Alt><Shift><F6> shall move focus forward and backward through the windows within a family. When a window receives focus, the window frame shall highlight and the window shall be raised to the front of the screen.

3.2 NAVIGATION WITHIN WINDOWS

At any one time, only one object (e.g., a control such as a text field or push button) in the window with input focus shall be able to receive input from the keyboard. The location cursor shall indicate the object with keyboard focus. When a window receives input focus, the location cursor shall be placed on the object that last had keyboard focus in the window. Users shall be able to move the location cursor between objects (i.e., navigate) within the window either with the pointing device or from the keyboard. Users shall also be able to select (and activate) an object using either input device.

3.2.1 Pointing Device Navigation for Controls

When users place the pointer on an object such as a control and click the Select button on the pointing device, the location cursor shall move to the object, and the object shall have keyboard focus. The shape of the location cursor depends on the type of object; in most cases, a box cursor will appear on the object. Using the pointing device to move keyboard focus to an object usually selects the object at the same time. To move keyboard focus without selecting an object, users shall place the pointer on the object, then hold down <Ctrl> while clicking the Select button on the pointing device.

Autoscrolling should be available when the pointer is on a scrollable control such as text or a list box. To autoscroll, users place the pointer on the control and press the Select button on the pointing device. When they drag the pointer outside the control, the control scrolls in the direction of the pointer. Users release the Select button to stop autoscrolling.

3.2.2 Keyboard Navigation for Controls

The Motif Style Guide divides the content of a window into tab groups in order to support keyboard navigation for controls. A tab group may contain a single control, such as a list box or text field, or a set of related controls, such as a set of radio buttons, check buttons, or push buttons. Navigation is provided between tab groups in a window as well as between controls within a group. The location cursor indicates the control with keyboard focus.

Application developers should provide capabilities for navigation between and within tab groups that fit the task being performed in the window and allow users to perform the task efficiently and without error. For example, text fields may be considered as individual tab groups, with <Tab> used for navigation between groups. Alternatively, sets of single-line text fields may be treated as a single tab group, and the arrow keys used to navigate among them.

3.2.2.1 Navigation Between Tab Groups

<Ctrl><Tab> shall move the location cursor to the next tab group in a window, and <Ctrl><Shift><Tab> shall move the cursor to the previous group. In addition, in all groups except multi-line text, <Tab> and <Shift><Tab> shall also move the location cursor forward and backward between tab groups.[16] If the tab group contains a set of related controls, the location cursor shall be placed on the default control (e.g., if the group is a set of push buttons where one has been designated as the default) or the first (i.e., top leftmost) available (i.e., selectable) control in the group. The direction of cursor movement within the window shall be from upper left to lower right and shall wrap between the first and last tab groups in the window. If none of the controls in a tab group can have keyboard focus (e.g., none is available for selection or activation), then <Tab> and <Ctrl><Tab> shall skip the group.

3.2.2.2 Navigation Within a Tab Group

<Up>, <Down>, <Left>, and <Right> shall move the location cursor between the controls within the tab group with keyboard focus. Moving the location cursor to a control shall not change the state of the control (i.e.., shall not move the highlight to the control). The direction of cursor movement shall be from left to right and top to bottom, as appropriate to the arrow key. The location cursor should wrap between the last and first control in a tab group unless the control is scrollable (see section 3.2.2.3).

The arrow keys shall move the location cursor one increment at a time (e.g., to the next line in text, to the next item in a list box). <Ctrl> in combination with the arrow keys shall increase the size of the increment (e.g., move the text cursor to the next word rather than the next character in text). In addition, <Home> shall move the location cursor to the leftmost element in the control, and <End> shall move the cursor to the rightmost element. <Ctrl><Home> shall move the location cursor to the top leftmost (i.e., beginning) element in the control, and <Ctrl><End> to the bottom rightmost (i.e., end) element.

3.2.2.3 Navigation in Scrollable Tab Groups

In controls such as text fields and list boxes that are scrollable, <PageUp>, <PageDown>, <PageLeft> (or <Ctrl><PageUp>), and <PageRight> (or <Ctrl><PageDown>) shall scroll the elements in the control up, down, left, or right one page minus one unit of information (e.g. one line of text) at a time, as appropriate. Keyboard focus remains on the element where it was positioned before the scrolling operation began, and the location cursor may no longer be viewable. However, any keyboard action that alters the element with keyboard focus or moves focus to another element shall scroll so that the element with focus is viewable.

3.2.2.4 Location Cursor Behavior During Keyboard Navigation

Developers should define tab groups within each window to support efficient navigation among sets of related controls. The identification of tab groups should be based on the order in which users are expected to interact with the controls in the window. A scroll bar should be included as a tab group so that users can move efficiently through the information in the scrollable area. Figure 3-1 presents a window in which the controls are divided into four tab groups: the set of radio buttons in Text Size, the text field in Text Size, the set of radio buttons in Symbol Size, and the set of push buttons at the bottom of the window.

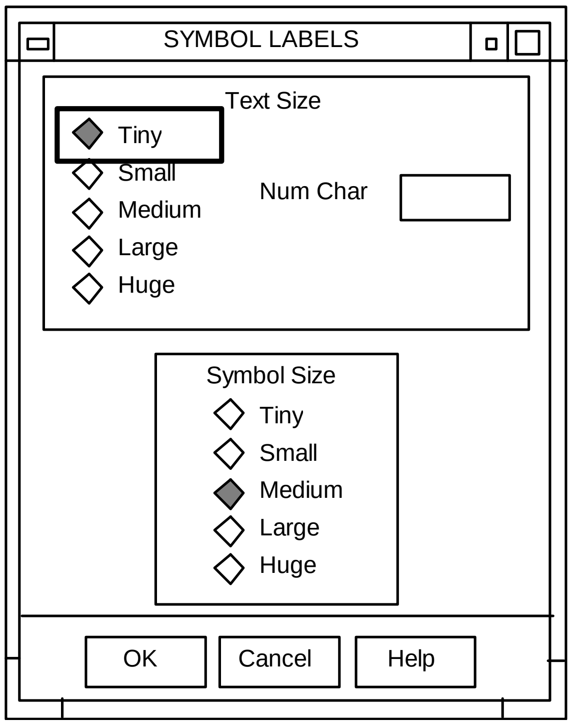


Figure 3-1. Example symbol label window (from the JMCIS style guide).

Developers should specify a default position for the location cursor in each window in the application. This position should be the control with which users are expected to interact first, and it is usually the top leftmost control on which the location cursor can be positioned. For example, when the window shown in figure 3-1 is displayed, the location cursor should be displayed on the Tiny radio button in the Text Size tab group. Whenever users open a window, a location cursor should appear in its default position in the window. One active location cursor shall be displayed in a window at any time, and this cursor shall indicate the location where keyboard input will occur.

When the location cursor moves into a tab group, the cursor should be placed on the first available control in the group. In figure 3-1, when users press <Tab>, the location cursor should move from the Tiny radio button in the Text Size tab group to the Num Char text field, to the Tiny radio button in the Symbol Size tab group, and finally to the OK push button. The location cursor should always be visible as it moves within a window; i.e., there should be no “invisible” tab groups which cause the location cursor to temporarily disappear as users navigate within the window. In addition, movement of the location cursor should be controlled from the keyboard, and the position of the location cursor should not be affected by movement of the pointer within the window. For example, the location cursor should not appear on a control within a tab group when the pointer moves into the group, and then disappear when the pointer moves out of the group.

Once the location cursor is positioned within a tab group, the arrow keys shall move the cursor between the available controls within the group. For example, pressing <Down> in the Symbol Size tab group should move the location cursor from Tiny to Small to Medium and so on. In the case of text fields, the arrow keys shall move the text cursor within the text in the field. For example, when the location cursor is on the Num Char text field (and a text cursor is displayed in the field), pressing <Right> should move the text cursor one character to the right.

When users move the location cursor to a control, the state of the control should not change. For example, when users move the location cursor between the radio buttons in the Text Size tab group, the selected state of the buttons should not change. Users must make a selection (e.g., by pressing <Space> or <Select>) while the location cursor is on a control in order to change the state of the control. If expert activation (see section 3.4.3.1 on selecting default actions) is available for a control, placing the location cursor on a control and pressing <Return> shall change the state of the control and execute the default push button action in the window.

3.3 OBJECT SELECTION

Application developers shall apply the selection methods defined in this section to the collections of objects that can be selected in a window. Motif defines three types of collections: a text collection (e.g., continuous text), a set of items in a list box, and a group of graphic objects.[17] Developers should identify the method, for each type of collection in the application, that allows users to make selections quickly and without error, and implement that method consistently whenever the collection is presented. The selection method(s) available for an object should match the type of action that can be executed on the object; for example, if a list box is provided in a window so that users can select a single item about which to obtain additional information, the selection method available in the list box should allow users to select only one item and not multiple items or none of the items. Developers should create alternate methods of selection only if none of the methods presented here provide the functionality required by the application. If an alternate method is created, it should conform to the methods in table 3-1 to the maximum extent possible.

When an element is selected, its appearance shall change to indicate the change in its state. Select state is usually shown by highlighting the element (i.e., change in color, foreground/background reverse video) but may be indicated in some graphic objects by the appearance of “handles” on the object. Clicking on an object with handles displays the handles which can also be used to reshape the object (by placing the pointer on one of the handles and then dragging it to a new position).

3.3.1 Pointing Device Selection Methods

Table 3-1 lists the methods available for selecting elements in a collection using the pointing device. In some cases, the methods vary slightly depending on whether the element is part of a one-dimensional collection such as text or a list box (i.e., where the elements can be treated as if they are in a line) or a two-dimensional collection such as a group of graphics objects (i.e., where an element has both a vertical and horizontal position within the collection). Application developers shall use the methods listed in table 3-1 for making selections in these types of collections and shall implement extended selection methods described in section 3.3.3.2 in text as appropriate.

[17]. The methods presented in these specifications address selection and deselection in collections that appear in a single window. Methods for selecting collections that appear in different windows will be addressed in a future version of this document.

Table 3-1. Pointing device selection methods in one-dimensional and two-dimensional collections.

To select one element:

Position the pointer on an unselected element and click the Select button on the pointing device. The location cursor moves to the element, the element is selected (and highlights), and any other element that was selected in the collection is deselected.

To select multiple elements one at a time:

Position the pointer on the first element and click the Select button to select it. Add elements to the selection by placing the pointer on other unselected elements and clicking the Select button on each of them. The location cursor moves to each element as it is selected. Place the pointer on a selected element and click the Select button to deselect it (and the element returns to its normal appearance).

To select a range of contiguous elements:

Position the pointer on the first element in the range to be selected, then press the Select button on the pointing device to set the anchor for the range. Any other element that was selected is deselected. Drag the pointer until it is on the last element in the range, and release the button to complete the selection. In one-dimensional collections: The elements within the range are selected (and highlight) as the pointer is dragged over them. In two-dimensional collections: As the pointer is dragged through the elements, a bounding box is displayed outlining the elements being selected. When the Select button is released, the box disappears and the elements that were in the box are selected (and highlight).

To extend a range selection (i.e., add/remove contiguous elements):

Position the pointer on the element that should be the last one in the selection, and hold down <Shift> while clicking the Select button on the pointing device. The elements in the revised selection range (defined from the original anchor to the current pointer position in one-dimensional collections, and defined by the diagonal from the anchor to the current pointer position in two-dimensional collections) are reselected (and highlight), and any elements removed from the selection return to normal appearance.

To add/remove a discontiguous element to/from a range selection:

Position the pointer on the element, then hold down <Ctrl> while clicking the Select button on the pointing device. If previously unselected, the element is selected (and highlights); if previously selected, the element is deselected and returns to normal appearance. The other elements in the selection remain highlighted.

To add/remove a discontiguous matrix of elements to/from a range selection:

Position the pointer on the first element in the range to be added/removed, then hold down <Ctrl> while pressing the Left button on the pointing device to set the anchor for the range. Any other elements that are selected remain selected. Drag the pointer until it is on the last element in the range, and release the button to complete the selection. In one-dimensional collections: If previously unselected, the elements in the range are selected (and highlight) as the pointer is dragged over them; if previously selected, the elements in the range are deselected and return to normal appearance. In two-dimensional collections: As the pointer is dragged through the elements, a bounding box is displayed outlining the elements. When the Select button is released, the box disappears. If previously unselected, the elements in the box are selected (and highlight); if previously selected, the elements in the box are deselected and return to normal appearance.

In addition to the methods in table 3-1, developers should provide users with the capability (e.g., as options in a pull-down menu or as push button actions), as appropriate, to quickly select and deselect all of the elements in a collection with the pointing device. If users are required to select an element from a group of closely overlapping elements, developers should implement a method (e.g., blink coding) whereby users can quickly and accurately select the element desired.

3.3.2 Keyboard Selection Methods

Two modes, normal and add, are available for selecting the elements in a collection using the keyboard. In normal mode, the location cursor and the highlight (indicating the current selection) are on the same element and move together when the arrows keys are used. In add mode, the location cursor moves independently of the highlight. The appearance of the location cursor shall provide an indication of the selection mode currently in effect. The shape of the location cursor shall be a solid rectangle in normal mode and a dotted rectangle in add mode.

Table 3-2 indicates how normal mode and add mode shall be used when selecting one or more elements in any collection (both one- and two-dimensional) except text. Add mode shall be used to select one element or multiple elements one at a time. Two methods are available for selecting multiple contiguous elements; developers shall implement the method based on normal mode and may also implement the method based on add mode. Both normal and add modes shall be used to select multiple discontiguous elements in a collection.

Table 3-2. Keyboard selection methods in collections other than text.

To select one element:

Use the arrows keys to move the location cursor to an unselected element. The location cursor moves independently of the highlight. <Space> (and <Select> if available) selects the element with the location cursor (and move the highlight to the element) and deselects any previously selected element in the collection.

To select multiple elements one at a time:

Use the arrow keys to move the location cursor to an element. The location cursor moves independently of the highlight. <Space> (and <Select> if available) selects (and highlight) an unselected element or deselects a selected element (and it returns to normal appearance).

To select a range of contiguous elements using normal mode:

Use the arrow keys to move the location cursor to the first element in the range to be selected. Press <Space> (or <Select> if available) to set the anchor for the selection. The element is selected (and highlights), and any other element that was selected is deselected. Use the arrow keys to move the location cursor (and highlight) to the last element in the range. <Shift><Space> (and <Shift><Select> if <Select> is available) extends the selection (and highlights all elements) from the anchor to the element with the location cursor. The location cursor remains on the last element in the range.

To select a range of contiguous elements using add mode:

Use the arrow keys to move the location cursor to the first element in the range to be selected. Press <Space> (or <Select> if available) to set the anchor for the selection. The element is selected (and highlights), and any other element that was selected is deselected. Use the arrow keys to move the location cursor to the last element in the range; the highlight remains on the anchor element. <Shift><Space> (and <Shift><Select> if <Select> is available) extends the selection (and highlight all elements) from the anchor to the element with the location cursor. The location cursor remains on the last element in the range.

To extend a range selection (i.e., add/remove contiguous elements) using normal or add mode:

Use the arrow keys to position the location cursor on the last element in the selection. Press <Shift> and use the arrow keys to move the location cursor to the other elements to include in/remove from the selection. Elements being added to the selection highlight as the location cursor moves to them; elements being removed from the selection return to normal appearance.

To add/remove discontiguous elements to/from a selection:

Select the elements in the contiguous part of the collection using normal mode (see above). Press <Shift><F8> to toggle to add mode. Select the element(s) in the discontiguous part of the collection using add mode (see above). Any elements that were selected using normal mode remain selected.

When a selection method involves both modes, <Shift><F8> shall toggle between them. <Ctrl></> shall be available, as appropriate, to select all of the elements in a collection and <Ctrl><\> to deselect all of the elements in a collection. In both cases, the location cursor remains at its current location. <Cancel> shall cancel or undo a selection action and return the elements to their normal appearance.

Because <Space> and <Shift><Space> are mapped to text entry (e.g., insert a space character in text), the Motif Style Guide provides an alternate set of key bindings for performing range selection in text collections. This document does not require that developers implement these key bindings unless this type of selection is integral to the functionality provided by an application and should be available to users even when operating a system in degraded mode. If an application provides this functionality, developers shall use the key bindings indicated in table 2-2 and 2-3 to do so.

3.3.3 Other Types of Selections

3.3.3.1 Default Actions

Motif defines two types of default actions: object activation (in which the default action displays a view of the object) and expert activation (in which the default action both selects an object and executes the action on it). When a default action is assigned to an object, the action shall be executed with the pointing device by double clicking on the object. In object activation, for example, double clicking on an icon shall restore it to a window. In expert activation, when users double click on an item in a list box (e.g., the name of a map), the item should be selected and the default action in the window (e.g., display the map) should be executed. In addition, after making a selection in a window, <Enter> or <Ctrl><Return> shall invoke the default action from the keyboard. If keyboard focus is on an object other than multi-line text, <Return> shall also execute the default action in a window.

Developers should use expert activation only as a short-cut to features available elesewhere in an application window. If this type of activation is available to users, the application should provide a means to reverse the effects of the action (e.g., by selecting an Undo menu option or a Cancel push button). Default actions should be defined for categories of objects (e.g., controls, text, lists, graphics); the operation should be the same for every object within the category and should be an action that users are most likely to execute on the object. For example, whenever users double click on a file icon, the file should be opened and a window with the contents of the file should be displayed.

3.3.3.2 Extended Selections

Users should be able to execute an extended text selection with the pointing device. When extended selection is available, clicking once should place the text cursor in the text, double clicking should select the word on which the cursor is placed, and triple clicking should select the entire line of text on which the cursor is placed.

3.3.3.3 Canceling a Selection

<Cancel> shall be available to stop or cancel the current action. <Cancel> in the dialog window with input focus shall be equivalent to selecting the Cancel push button in the window. <Cancel> shall cancel any autoscrolling action performed with the pointing device and return the slider to its position prior to the start of scrolling. <Cancel> shall dismiss a pull-down menu without making a selection. <Cancel> while pressing the Select button on the pointing device shall cancel a selection action being performed with the pointing device.

3.4 OBJECT TRANSFER

Users should be able to transfer an object or collection of objects (such as individual data records or entire files) to a new location in the same window, to a different window in the same application, or to a window in a different application. This section describes the four transfer methods supported by Motif: drag transfer, clipboard transfer, primary transfer, and quick transfer.

3.4.1 Drag Transfer

Drag transfer allows users to move, copy, and link objects by dragging them from one location to another.[18]

To perform a drag move, users shall place the pointer on an object, then hold down <Shift> while dragging the object using the Transfer button on the pointing device. When the Transfer button is released, a copy of the object is pasted at the location of the pointer, and the original object is deleted.

To perform a drag copy, users shall place the pointer on an object, then hold down <Ctrl> while dragging the object using the Transfer button on the pointing device. When the button is released, a copy of the object is pasted at the location of the pointer, and the original object remains at its original location.

To perform a drag link, users shall place the pointer on an object, then hold down <Ctrl><Shift> while dragging the object using the Transfer button. When the button is released, a link is created from the destination to the object at its original location.

The default shall be to use the Transfer button to do a drag transfer and the result shall be a move. <Cancel> shall cancel a drag operation that is in progress and return the object being dragged to its location before the start of the drag operation.

[18]. These specifications describe drag transfer functionality available in Motif 1.2. Developers working with earlier versions of Motif should implement only the functionality available in the version they are using, rather than attempt to provide the full range of functionality described here.

Drag transfer can be used to transfer an unselected element in a collection, a set of selected elements in a collection, or an entire collection. When a set of selected elements is moved within the same component, the elements remain selected after they have been moved. In general, initiating a drag transfer in a set of selected elements shall drag the entire selection, even if there are discontiguous elements in the selection. In collections such as lists and graphics, initiating a drag transfer on an unselected element shall drag only the element and not affect any other elements in the collection that may be selected. In addition, if a drag is initiated at a location where there are multiple overlapping elements in these types of collections, the drag shall occur on the highest draggable element in the stacking order.

During the drag operation, the shape of the pointer shall change to a drag icon, such as the ones shown in figure 3-2, to indicate that a drag is in progress. When the transfer is complete, the drag icon shall change back to a pointer. The drag icon shall contain a source indicator showing the type of object (e.g., text, graphics) being dragged. The icon may also contain an operations indicator showing the type of drag operation (e.g., move, copy, link) being performed and a state indicator showing whether the current position of the icon is a valid drop site for the element(s) being dragged. Application developers should use the drag icons shown in figure 3-2 to indicate move, copy, and link operations for text and graphics objects. Application developers can define new drag icons as needed to represent drag operations not included in figure 3-2. When doing so, developers should follow the rules indicated in section 2.2.2 on defining new pointer shapes.

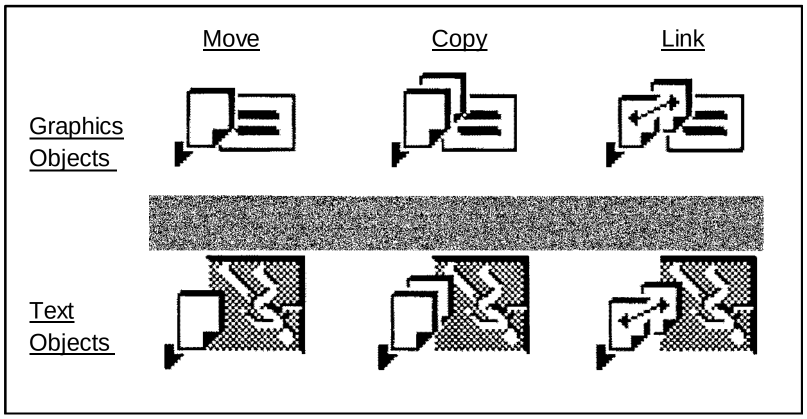


Figure 3-2. Drag icons for move, copy, and link operations (from the Motif Style Guide).

Whenever possible, drag transfer should be available within an application so that users are able to transfer elements within and between collections in the application. If appropriate, drag transfer may be provided only on a limited basis (e.g., drag only, drop only) depending on the type of collection. Applications should also provide the capability to transfer elements between collections in other applications. Each system in which an application is installed should define the extent to which this capability will be available to users. To support consistent implementation across applications, developers providing a drag transfer capability should do so according to the specifications provided in this document.

3.4.2 Clipboard Transfer

Clipboard transfer allows users to move, copy, and link objects by transferring them first from their current location to a temporary clipboard and then from the clipboard to a new location. Clipboard transfer can be used, for example, to transfer text (e.g., strings of characters, blocks of text) and graphics (e.g., lines, shapes, entire figures) within a window and from one window to another in the same or different applications.

A clipboard move consists of cut and paste operations, while a clipboard copy and a clipboard link consist of copy and paste operations.

Cut should clear the clipboard, store a copy of the object in the clipboard, and remove the object from its original location. If the object being transferred is graphic, the space that it occupied should be left blank. If the object is text, the remaining text should be compressed (i.e., shifted to the left) to fill in the space.

Copy should clear the clipboard and store a copy of the object being transferred in the clipboard. The object should remain at its original location.

Paste should copy the object in the clipboard to the new location. If the object is graphic, the paste operation should copy it to the location with keyboard focus. If the object is text, the paste operation should copy the object to the current position of the text cursor; the text should appear to the left of the cursor and any existing text should move to the right. The object should remain in the clipboard so that a copy of the object can be pasted whenever this command is selected. Pasting an object at a location should not normally select the object; i.e., its appearance should remain unchanged when it is transferred from the clipboard.

A clipboard link allows users to place a link in the clipboard to selected objects and then to transfer the link from the clipboard to a new location. A link can be placed at this location by executing a Paste or Paste Link action. Motif does not define a mnemonic or keyboard accelerator for a Copy Link or Paste Link operation.

Clipboard transfer operations should be available whenever an object that can be edited has keyboard focus. Applications should provide access to these operations as menu options or push buttons in a window, and users should be able to execute the operations either with the pointing device or from the keyboard. The Motif default bindings and key mappings for Cut, Copy, and Paste operations are given in tables 2-2 and 2-3, respectively, and the keyboard accelerators and mnemonics for these and other clipboard transfer operations are listed in table 5-1.

Developers should provide access to clipboard transfer in a consistent fashion (e.g., using the vocabulary, mnemonics, and keyboard accelerators presented here) throughout the application. In addition, developers should provide access to clipboard transfer only when it is appropriate to the task being performed by users. For example, developers may provide access to clipboard transfer when users are working in a word processing window but may consider these operations inappropriate to implement in the text fields within a data entry window (e.g., to prevent users from pasting text into a field that is not long enough to accept it). Finally, users should be able to view the contents of the clipboard and should be informed (e.g., in a message window) when they attempt to cut or copy an object whose size exceeds the capacity of the clipboard.

3.4.3 Primary Transfer

Primary transfer allows users to transfer a selection directly to a destination without using the clipboard for intermediate storage.

To perform a primary move, users shall select an object, move the pointer to the destination, and press <Shift> and click the Transfer button to move the object to that location.

To perform a primary copy, users shall select an object, move the pointer to the destination, and press <Ctrl> and click the Transfer button to copy the object to that location.

To perform a primary link, users shall select an object, move the pointer to the destination, and press <Ctrl><Shift> and click the Transfer button to create a link between the object and the location.

Transferring an object by performing a primary copy or a primary link shall not select the object; however, transferring an object via a primary move shall do so. When the Transfer button is used to perform a primary transfer, the default shall be a copy operation. Users can also execute a primary copy and a primary move from the keyboard, using the Motif default bindings listed in table 2-2.

If an application supports primary transfer, it shall be implemented as defined here. The keyboard accelerators in table 5-1 should be used if access to primary transfer operations is provided from a pull-down or pop-up menu within an application.

3.4.4 Quick Transfer

Quick transfer allows users to temporarily select an object and immediately transfer it to a new location. Whereas primary transfer is available for editable objects, quick transfer can be used to transfer static text (e.g., the label for a text field or a push button) or graphics that are not normally selectable.

To perform a quick move, users shall place the pointer on an object, hold down <Alt><Shift> while dragging the object using the Transfer button. The object is temporarily selected and, when the Transfer button is released, moved to the new location.

To perform a quick copy, users shall place the pointer on an object, then hold down <Alt><Ctrl> while dragging the object using the Transfer button. The object is temporarily selected and, when the Transfer button is released, copied to the new location.

To perform a quick link, users shall hold down <Alt><Ctrl><Shift> while dragging the object using the Transfer button. The object is temporarily selected and, when the Transfer button is released, linked to the new location.

When <Alt> and the Transfer button are used to perform a quick transfer, the default shall be a copy operation. Using quick transfer shall not select the object being transferred.

3.5 INTERACTIVE CONTROL

3.5.1 Object-Action Selection

Applications should base their interactions with users on an object-action selection paradigm in which users first select an object of interest (so that it has keyboard focus) and then select an action to perform on that object. More than one object may be selected, and the objects may be controls (e.g., radio or check buttons), text (e.g., individual characters, strings of characters), or graphic (e.g., track symbols on a map), and the action may require a single selection by users (e.g., select a command from a pull-down menu) or multiple selections (e.g., display a dialog window, select option(s) in the window, and select an action to execute). Users should be informed (e.g., with feedback in the message bar of a window) when an application requires interactions that diverge from an object-action selection paradigm.

3.5.2 User Control of Interaction

Users should control the pace of the interaction with an application and should not be forced to interact with an application at a specified rate. The application should execute an action only in response to explicit user input. However, the pace of user input should not slow down the speed of application processing or execution.

3.5.3 Immediate Feedback

When users interact with an application, they should receive immediate feedback on the result of their action. Application developers should ensure that when users take an action (e.g., select a push button), there is an immediate and visible response to the action (e.g., the button highlights and the action indicated on the button is executed). A visible response to the action should occur even if the result of the action cannot be displayed immediately (e.g., the pointer should change to a watch or a message should appear in the window while processing takes place).

An application should inform users when they can and cannot take an action. An application should provide visual cues indicating when it can accept input (e.g., the pointer appears as a shape supporting selection), when it is temporarily unavailable (e.g., the pointer appears as a watch), and when it is unavailable during extended processing (e.g., a working window appears on the screen). In addition, the appearance of an object should provide an indication of its availability. For example, the menu options in an application should provide feedback concerning the actions that can be performed on selected object(s). If a menu option is not appropriate for an object that has been selected or if a menu option cannot be executed for some reason, users should not be able to select the menu item and a visual cue (such as graying the option) should be provided.

When users must perform several actions to complete an operation, an application should prompt users with the actions that need to be performed. For example, when users select a Zoom Area option from a pull-down menu, the application might provide feedback explaining how to select the area to be zoomed (e.g., a message at the bottom of the window indicating "Click on the map, then drag the pointer over the area to be zoomed.").

An application should ignore user actions made during periods when input cannot be accepted. For example, an application should accept keystrokes made by users only when the text cursor appears in the window with input focus; if the cursor is not present in the window, the keystrokes should be ignored by the application. In addition, an application should disable the pointing device and/or keyboard when input may have destructive effects (e.g., when the watch or caution pointer is displayed, indicating that the application is temporarily unavailable or that input cannot be accepted at the location of the pointer). Disabling is particularly important so that any input made by users during processing by the application is not stored and then executed when the application becomes available again. Although an application should not allow users to override disabling, users should be able to stop a process if desired (e.g., by selecting a Cancel push button).

3.5.4 Response Time

Users should receive an immediate indication that their action has been accepted by the application. MIL-HDBK-761A recommends that some visible response should be made within 0.2 seconds of any user action (e.g., a checked button is highlighted, a push button is highlighted) and the response to simple requests for data display should take no longer than 0.5 to 1.0 seconds. Responses to requests for new displays may take from two to ten seconds, especially if the operation requires more complicated operations such as accessing different files or transforming data. The DoD style guide indicates that system response time (i.e., the time between keystroke and screen response) should be in the range of 5 - 50 milliseconds and no longer than 0.2 seconds. These guidelines should be considered as recommended response times; it is expected that actual response times will be determined by factors such as the hardware configuration being used, the size of the track database being maintained, and the amount of communications processing being performed.

Error feedback (e.g., when a user attempts an action that is invalid) should be provided to users within two seconds of the time the error was detected. In addition, users should receive feedback indicating that their action is being processed by the application. When a user's request requires more than two seconds to process, an application should change the pointer shape to a watch. When a user's request requires more than five seconds to process, an application should display a message window stating that a lengthy operation is underway and providing an indication as to the progress of the operation being performed. This type of feedback informs users that the application is not responding because it is processing their previous input; as a result, users will not continue to make selections that are executed when the application becomes available again.

3.5.5 Error Detection

If users attempt to execute an invalid action, the application should not execute the action except to display a message describing the action that is invalid. If users make multiple errors within a single action, they should be notified of each occurrence of an error. The feedback should be immediate, be visual and/or auditory, and explain the nature of the error made. For example, if users attempt to enter alphabetic characters when only numbers are valid, the application should provide feedback (e.g., display an error window) that informs users of the error. If users repeat an error, the feedback should be different (perhaps auditory with diagnostics or help also provided) to show users that their attempted corrective action was, in fact, processed. Users should be required to correct only the invalid action and not to repeat the entire action sequence. After correcting the error, users should execute the same action (e.g., select a push button) for re-entry that was used for the original entry.

3.5.6 Explicit Destruction

Applications shall be designed so that users must confirm destructive actions (i.e., actions with irreversible negative consequences such as deleting a data file) before the action is executed by the application. Users shall have to respond to a confirmation prompt whenever they select a destructive action that cannot otherwise be reversed or "undone." If the destructive action applies to more than one object (e.g., multiple files), the prompt should list the objects and allow users to select the one(s) to which the action should be applied.

Users should not have to confirm the action when they close a window unless the action will cause significant data loss. In general, closing a primary window is a potentially destructive action because this type of window usually generates data that should not be lost as a result of such an action. Closing a secondary window is usually nondestructive because these windows are used for obtaining additional input from users that does not have to be saved.

When a confirmation prompt is displayed, the window in which the destructive action was taken should remain displayed until users make a selection to confirm the action. The window should not be removed when the prompt is presented and then redisplayed if users fail to confirm the destructive action.

3.5.7 General "Undo" Capability

Applications should be designed so that users can "undo" the selection or action most recently made unless the selection was one requiring explicit destruction. In addition to being able to "undo" commands such as cut, copy, and paste, users should be able to deselect objects, return an object to its prior state before an action was executed, and retrieve information that was removed from the screen. If applications cannot provide an "undo" capability, they should label irreversible actions as such and clearly separate actions that are reversible from those that are not.

3.5.8 Use of Processing Modes

Processing modes are system states where user actions have different results depending on the mode currently in effect. Motif uses modes when defining how users interact with some elements of the user interface. For example, Motif assigns modes to dialog windows (see section 4.4.2.1) that determine the extent to which users can interact with other windows while the dialog window is displayed. Similarly, Motif defines modes for text entry (i.e., insert and replace), with <Insert> used to toggle between them. Finally, menus are modal; i.e., when one is displayed, users cannot interact with other elements of the application.[19]

Developers should avoid using processing modes in their applications whenever possible. Instead, an application should be designed so that the same action has the same effect whenever it is executed by users. If a mode is used, the application should provide a visual cue to indicate the mode currently in effect. For example, when a palette (e.g., with drawing tools) is included in an application (see section 8.1.3.4), users should select one of the tool buttons to invoke the mode (e.g., drawing a square), and that button should remain highlighted to indicate the mode in which the user is operating. In addition, the application should be designed to minimize the extent to which user interaction is limited by the modes defined by Motif. For example, because menus are modal, they should not contain options that users may want to have available while interacting with other parts of the application. Alternatively, adding a tear-off button to a pull-down menu allows the menu to remain available after users make a selection; selecting the tear-off button changes the menu into a dialog window with the same contents as the menu, thus allowing users to make multiple selections.

3.6 CONSISTENCY IN PERFORMING OPERATIONS

3.6.1 Consistency at the Application Level

Application developers should identify the set of operations that are supported within the application and then specify the sequence of actions that users follow to perform each operation. This sequence should allow users to complete the operation rapidly and with a minimum number of actions. Developers should ensure that each operation can be performed by following its standard action sequence which is the same throughout the application. Developers can provide other approaches to performing an operation (e.g., to accommodate differences between novice and expert users) but should ensure that the standard sequence for the operation is always available to users.

---

[19]. In some cases, Motif does not provide a visual cue of the mode currently in effect. For example, dialog windows do not provide a visual indication of their modality, and the shape of the text cursor does not change to indicate the text entry mode currently in effect.[20]. Although the Motif Style Guide indicates that secondary windows shall not have a Minimize button, the specifications presented here allow for this component to be available in order to provide greater flexibility in window management (e.g., so that users have easy access to window functionality without obscuring the geographic display).

When defining an action sequence, developers should ensure that users have to execute only those actions that are required to perform an operation. For example, if an application requires users to enter location information for a contact by clicking the position on a map, the application should record the position and use the information; users should not have to type the same position information in a later part of its dialog with the application. If the system does not record the position selected, then users should not have to perform the action (i.e., click on a position). Similarly, when defining an action sequence, developers should minimize the frequency with which users have to move between input devices as they perform a task. For example, a data entry window should be designed so that users do not have to move their hands back and forth from the keyboard to the pointing device to enter values in the window. At a minimum, if the window contains an OK push button, it should be designated as the default so that users can keep their hands on the keyboard during data entry and then press <Enter> to commit the information.

3.6.2 Consistency at the System Level

Each system should identify the set of operations that are performed across multiple applications and then specify the manner in which these operations will be executed within the system. For example, a system might specify that whenever an application presents a set of records (e.g., tracks or messages) containing sortable fields, users will sort the records by clicking on the title of the data field. Users should be able to sort records by executing this action whenever they perform this operation in any application within the system. Users should not have to perform the same operation in different ways depending on the application being accessed (e.g., by selecting a Sort push button or selecting the data field as an option in a pull-down menu).

4.0 WINDOWS

4.1 WINDOW COMPONENTS

4.1.1 Standard Components

The standard window, shown in figure 4-1, consists of (1) a window frame containing a title bar, resize border, Window Menu button, and window control buttons and (2) an application or client area within the frame for an application to display information and interact with users. The title bar contains a Window Menu button at the left edge and Minimize and Maximize buttons at the right edge; the name of the window appears in the middle of the title bar. The application area can be partitioned into multiple work areas to present different kinds of information.

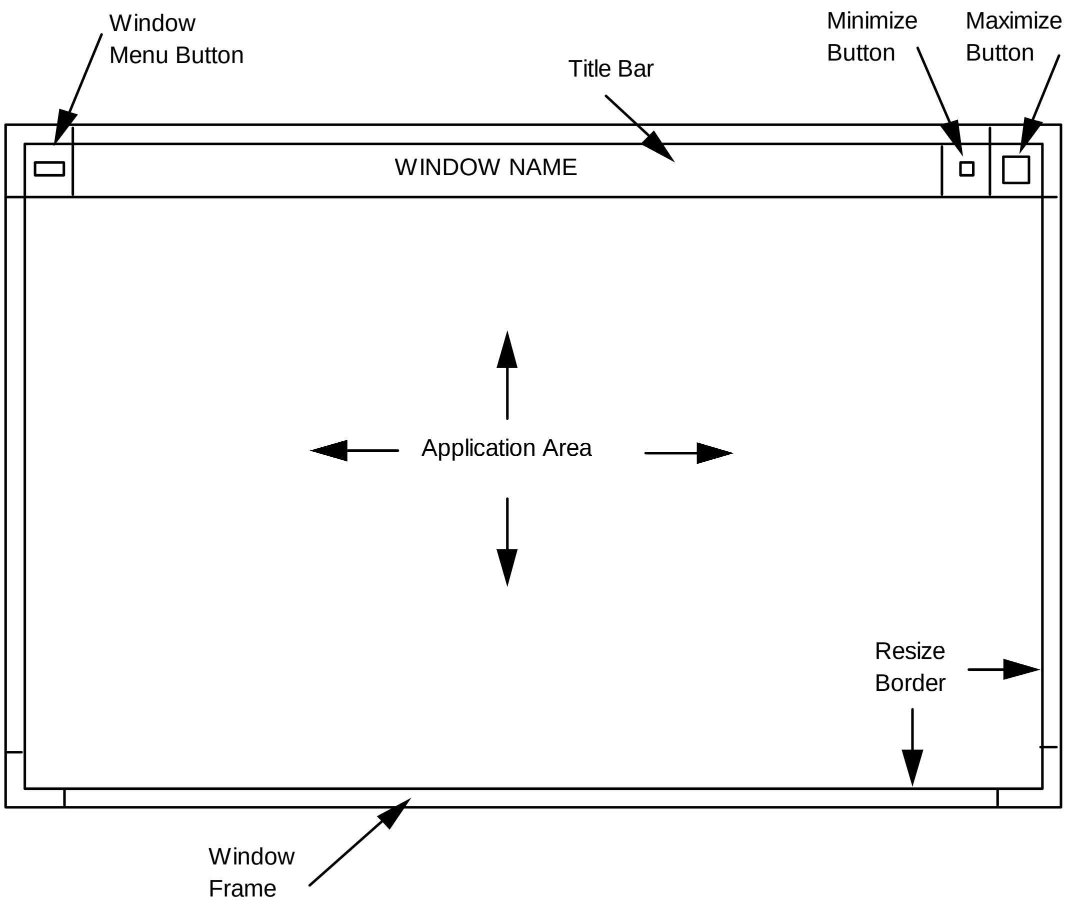


Figure 4-1. Standard components of a window (from the JMCIS style guide).

The window frame contains components that provide access to standard window management functions. If additional window management functions are needed, they should be mapped to buttons that are placed to the left of the Minimize button or the right of the Window Menu button. In addition, each button action shall be included as an option in the Window Menu.

4.1.2 The Window Menu

4.1.2.1 Appearance

The Window Menu button provides access to a menu containing the following window management options:

Restore	Restores a minimized or maximized window to its normal size. This function is not available (i.e., the item is dimmed) when the window is normal size.
Move	Moves a window on the screen.
Size	Changes the size of the window in the direction indicated by the pointer.

Minimize	Changes a window into an icon.
Maximize	Enlarges a window to its maximize size.
Lower	Moves a window to the bottom of the window hierarchy.
Close	Closes a window and removes it from the screen.

The Window Menu in a primary window shall contain all of the window management functions listed above. The Window Menu in a secondary window shall contain the options corresponding to the window components included in the window. If Lower is included in the Window Menu of a secondary window, selecting this option shall lower the window and all other windows that are children of the window's parent window (see section 4.4 on types of windows). The Window Menu in a menu window shall contain Move, Lower, and Close and shall not include Restore, Size, Minimize, or Maximize.

4.1.2.2 Behavior

Spring-loaded and posted menu selection methods (see section 5.1.2.1) shall be used to display the Window Menu and select options with the pointing device. Double clicking on the Window Menu button shall execute the Close option in the menu (see section 4.3.1.8).

<Shift><Escape> or <Alt><Space> shall select the Window Menu button, display the Window Menu and move the location cursor to the first available option in the menu. User shall move the cursor and select an option as they would in any pull-down menu. In addition, users can execute mnemonics or keyboard accelerators to select options in the Window Menu (see section 5.5 and 5.6). <Cancel> shall dismiss the menu without making a selection and return the location cursor to its position before the Window Menu was displayed.

4.1.3 Components of Primary and Secondary Windows

A primary window shall contain all of the standard window components shown in figure 4-1. A secondary window shall contain a title area and a Window Menu button. A secondary window may also contain resize borders and a Maximize button if users should be able to resize the window, and a Minimize button if users should be able to minimize the window.[20] Users should be able to move a dialog window (e.g., a message or prompt window) but not to minimize, maximize, or resize it unless this flexibility is called for by operational requirements.

Application developers should ensure that each window in the application contains only those components that are appropriate for the window type and that the components in each window behave in a consistent manner. For example, selecting a Minimize button should always change the window into an icon. Each window should contain only one set of window components that are selectable by users; if an application incorporates windows that have been developed using a toolkit other than Motif, the application should integrate these windows (e.g., with a Motif window frame) so that only one set of components is available for window manipulation.

4.2 WINDOW STATES

4.2.1 Open

A window is open when it is displayed on the screen. Multiple windows may be open at any one time and may partially or totally obscure one another on the screen.

4.2.2 Active

A window shall be considered to be active when it has input focus. A window shall have input focus when it first appears on the screen. A window that loses input focus (i.e., when users click in a different window) continues to be displayed on the screen but may be obscured by other windows.

4.2.3 Minimized

A window that does not need to be kept open may be minimized. When a window is minimized, it shall be replaced with an icon, and any processing that was occurring in the window shall continue.

4.2.4 Closed

When a window is closed, it shall be removed from the screen. In most cases, all processing relating to the window shall stop. Exceptions include windows where background processes, message transmission, and database updates are occurring.

4.3 WINDOW MANAGEMENT FUNCTIONS

4.3.1 Pointing Device Interaction

4.3.1.1 Placing a Window

Application developers shall define the normal, maximum, and minimum size of each window in the application and specify the location for each window when it first appears on the screen. Users may move and/or resize a window that includes these components. When a window is closed and then reopened, it should be displayed at the same location where it was when it was closed.

4.3.1.2 Moving a Window

Two methods shall be available for moving a window using the pointing device. With the first method, users shall select the Move option in the Window Menu. The four-directional arrow pointer shall appear in the middle of the window. As users move the pointer, an outline of the window shall move on the screen. An indicator showing the current location of the window in pixels shall also be displayed. Users shall click the Select button on the pointing device to place the window in its new location (and the pointer shall return to its normal shape).

With the second method, users shall position the pointer in the title bar, press the Select or Transfer button on the pointing device, and then drag the window to its new location. An outline of the window shall move on the screen and a location indicator shall be displayed as users move the pointer. Users shall release the button on the pointing device to place the window in its new location.

4.3.1.3 Resizing a Window

Two methods shall be available for changing the size of a window (not necessarily to its full size) using the pointing device. With the first method, users shall select the Size option in the Window Menu. The four-directional arrow pointer shall appear in the middle of the window. Users shall move the pointer to the frame border of the window. When the pointer reaches the border, it shall change to a resize pointer. As users move the pointer, an outline of the window shall move on the workspace, and an indicator showing the current dimensions of the frame shall be displayed. When users click the Select button on the pointing device, the window shall be redrawn to the new size (and the pointer shall return to its normal shape).

With the second method, users shall position the pointer on the resize border, and the pointer shall change to a resize pointer. Users shall then press the Select or Transfer button on the pointing device and drag the border to change the window size. An outline of the window shall move on the screen and a frame size indicator shall be displayed as users move the pointer. The window shall be resized when users release the button on the pointing device (and the pointer shall return to its normal shape).

4.3.1.4 Minimizing a Window

To minimize (i.e., iconify) a window, users shall either select the Minimize option in the Window Menu or click on the Minimize button on the title bar. In both cases, the window is minimized, and the icon appears in the lower left corner of the screen.

To open an icon (i.e., restore a window that has been minimized), users shall position the pointer on the icon and double click the Select button on the pointing device. When an icon is opened, the window shall be displayed at the front of the screen, at the same location where it was before being minimized.

4.3.1.5 Maximizing a Window

To expand a window to its full size, users shall either select the Maximize option in the Window Menu or click on the Maximize button in the title bar. In both cases, the window automatically expands to its full size. To return a maximized window to its normal size, users shall either select the Restore option in the Window Menu or click on the Maximize button in the title bar. In both cases, the window automatically returns to its normal size.

4.3.1.6 Raising a Window

To raise a window, users shall click on the title bar of the window. The window moves to the top of the window hierarchy on the screen and receives input focus.

4.3.1.7 Lowering a Window

To lower a window, users shall select the Lower option in the Window Menu. The window moves to the bottom of the window hierarchy on the screen.

4.3.1.8 Closing a Window

To close a window, users shall either select the Close option from the Window Menu or double click on the Window Menu button. If processing is occurring or unsaved data have been generated in the window, users shall be required to confirm the action before the window is removed from the screen and processing stops.

4.3.2 Keyboard Interaction

When users perform window management functions from the keyboard, they do so through the Window Menu. Keyboard interaction with the Window Menu is described in section 4.1.2.2.

4.4 TYPES OF WINDOWS

4.4.1 Primary Windows

Motif assumes that an application uses a primary window to conduct most of its interactions with users. When additional information is needed, a secondary window such as a dialog window[21] is displayed to carry out context-specific dialogs. When the dialog is complete, the secondary window disappears from the screen. In figure 4-2, window 1 is a primary window, and windows 2 and 3 are secondary windows.

[21]. These specifications refer to dialog windows and message windows (rather than dialog boxes and message boxes, as is done in the Motif Style Guide) to reduce possible confusion with window controls such as list boxes.

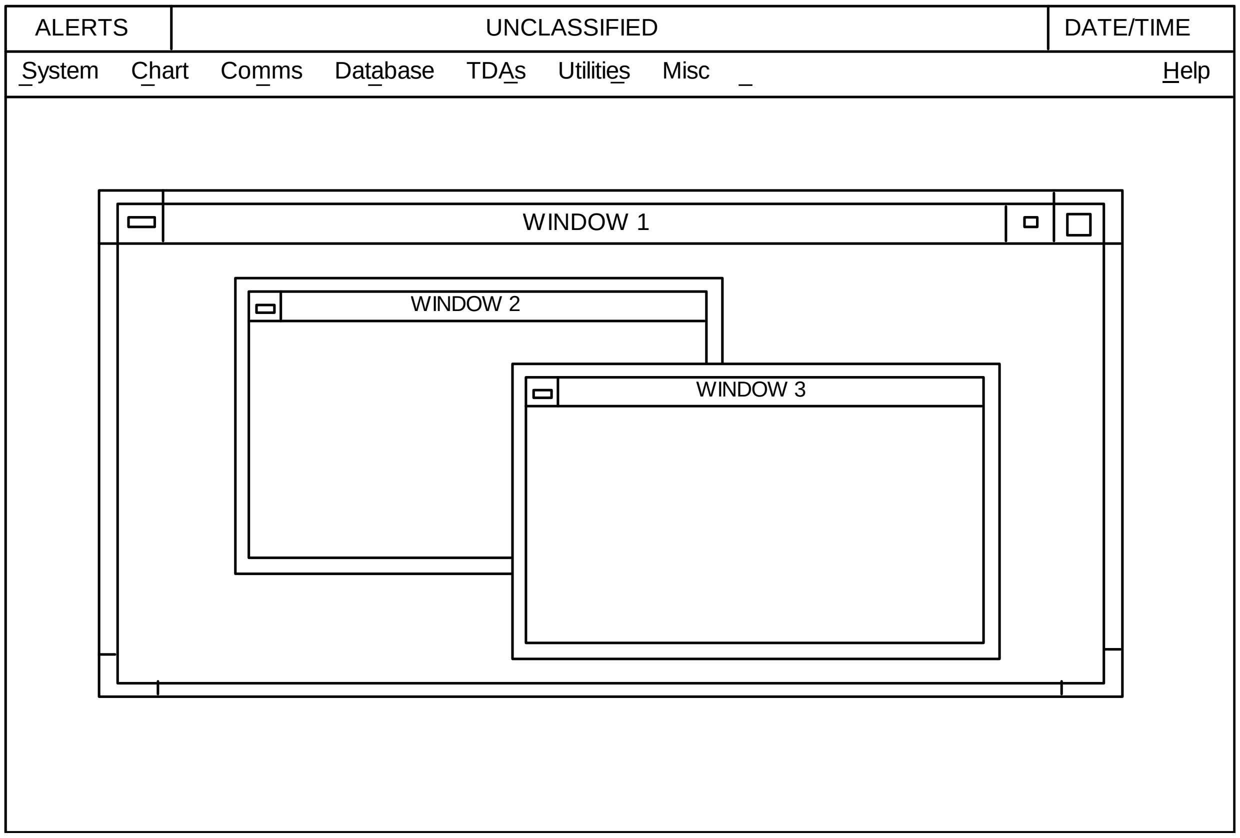


Figure 4-2. Example primary and secondary windows (from the DoD style guide).

The specifications presented here assume that an application can have more than one primary window (e.g., a map primary window, a database primary window, a message generation primary window) depending on the functionality provided. The specifications also assume that an application may provide access to its functionality either directly (i.e., through a primary window such as a map) or indirectly from within another application (e.g., a tactical decision aid that is executed only when contact data are displayed on a map). In the latter case, the primary window for one application (i.e., the decision aid) may have the appearance of a secondary window in the map application.[22]

Application developers shall define a hierarchy of primary and secondary windows that provide the functionality called for by the application. A family of windows consists of a primary (or parent) window and all of the secondary (or children) windows that are subordinate to the primary window. A primary window can have multiple secondary windows that are children

[22]. To minimize confusion (given that a primary window as defined by the application may have the appearance of a secondary window as viewed by the user), all references to primary and secondary windows in this document assume the developer's perspective, unless otherwise indicated.

to the parent window. Also, a secondary window may be the parent of one or more other secondary windows.

When a primary window is minimized, it and all of the secondary windows for which it is the parent shall be removed from the screen and replaced with an icon representing the primary window. All processing in the primary window shall continue. When the icon for the window is opened, the window and all of its secondary windows that were displayed on the screen when the window was minimized shall re-appear. Each primary window in an application shall be iconifiable separately.

When a primary window is raised or lowered, it and all of the secondary windows for which it is the parent shall move with it. When a primary window is closed, it and all of the secondary windows for which it is the parent shall be removed from the screen, and all processing in the primary window shall stop (except as noted in section 4.2.4).[23] When the last primary window for an application is closed, the application shall also be closed.

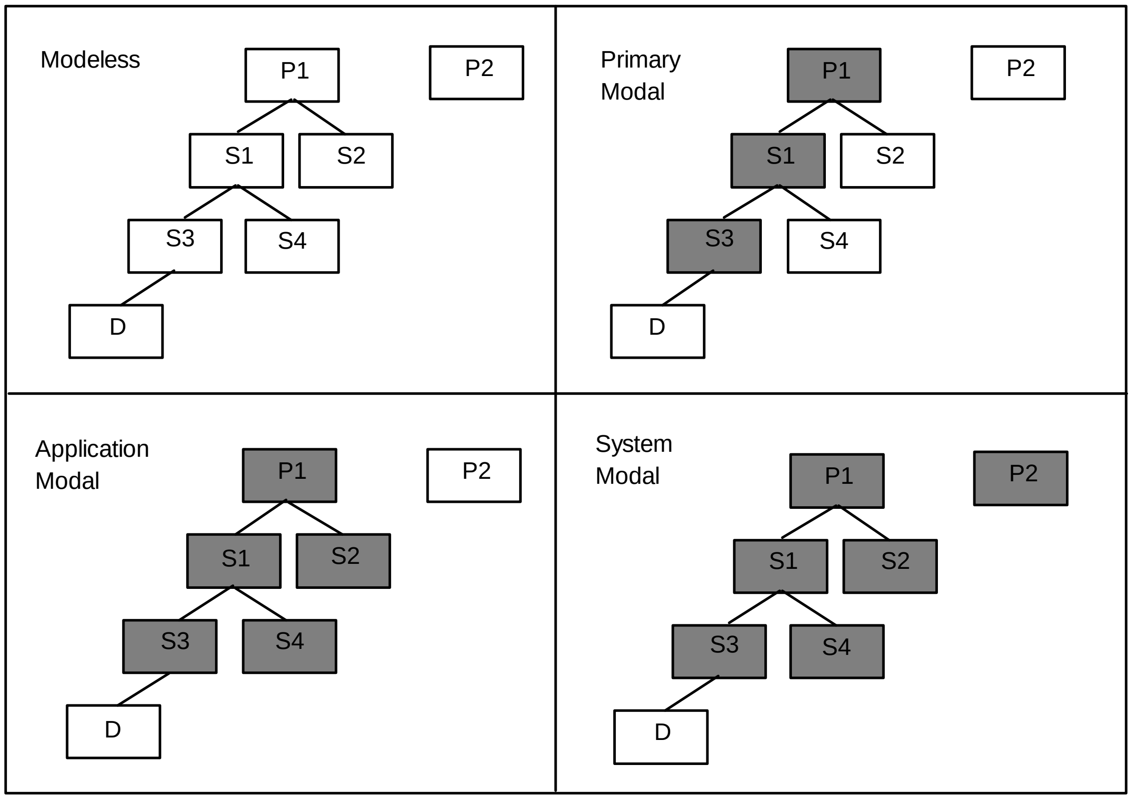
4.4.2 Secondary Windows

When a secondary window is opened, it shall appear in front of its parent window, and the parent window shall remain displayed. When a secondary window is closed, it and any other windows that are its children shall be closed but its parent window shall not be affected.

4.4.2.1 Modes of Interaction

The mode assigned to a dialog window determines the extent to which users can interact with other windows while the dialog window is displayed. If a dialog window is modeless, users shall be able to interact with other windows while it is displayed on the screen. If a dialog window is modal, users shall be restricted from interacting with other application and/or system windows while it is displayed. Motif defines three types of modal windows: primary modal (i.e., does not allow interaction with any parent of the window), application modal (i.e., does not allow interaction with any window created by the same application even if the application has multiple primary windows), or system modal (i.e., does not allow interaction with any other window on the screen). Figure 4-3 illustrates each of these types of modality.

[23]. The behavior of a secondary window that is a child to multiple primary windows will be addressed in a future version of this document.



Key:
P = Primary Window
S = Secondary Window
D = Dialog Window

■	= Input Locked
□	= Input Open

Figure 4-3. Interaction restrictions in modeless and modal windows (from the Operations Directorate style guide).

4.4.2.2 Dialog Windows

Developers should use dialog windows with the lowest necessary level of modality so that users are interrupted as little as possible while working in the application. Prompt windows (see section 8.2.1), for example, should be modal. A message window (see section 8.2.2) may be modal or modeless depending on its purpose. Information message windows shall be modeless, while error, question, and warning message windows should be modal. Working message windows should also be modal, unless the window is referring to another device (e.g., a printer) that is working, in which case the window should be modeless. A command window (see section 8.2.3) shall be modeless. A selection window (see section 8.2.4) may be modal, while a file selection window should be modeless.

4.4.2.3 Menu Windows

A menu window is a secondary window containing a menu, usually one that has been torn off. A menu window may be the child of a primary window, a secondary window, or another menu window. A menu window should be modeless so that users can select options from the window while interacting with other windows that are displayed on the screen.

4.5 WINDOW PLACEMENT

Windows can be displayed in either an overlapping or tiled arrangement on the screen. With overlapping placement, windows are displayed on top of one another, as shown in figure 4-2. When a new window is opened, it is displayed in front of those already on the screen. Because multiple windows can be stacked on each other, a window may be partially visible or completely obscured by other windows that overlap it. With overlapping placement, window management functions (e.g., minimize, maximize, resize) are available to allow users to control the size and position of individual windows so that the screen can be organized to support the particular task being performed. By contrast, with tiled placement, as shown in figure 4-4, windows are sized and positioned so that each one is completely visible at all times. Window management options are usually restricted so that users have little flexibility to modify the arrangement defined by the system; i.e., the windows in a tiled placement cannot be moved, resized, or closed.

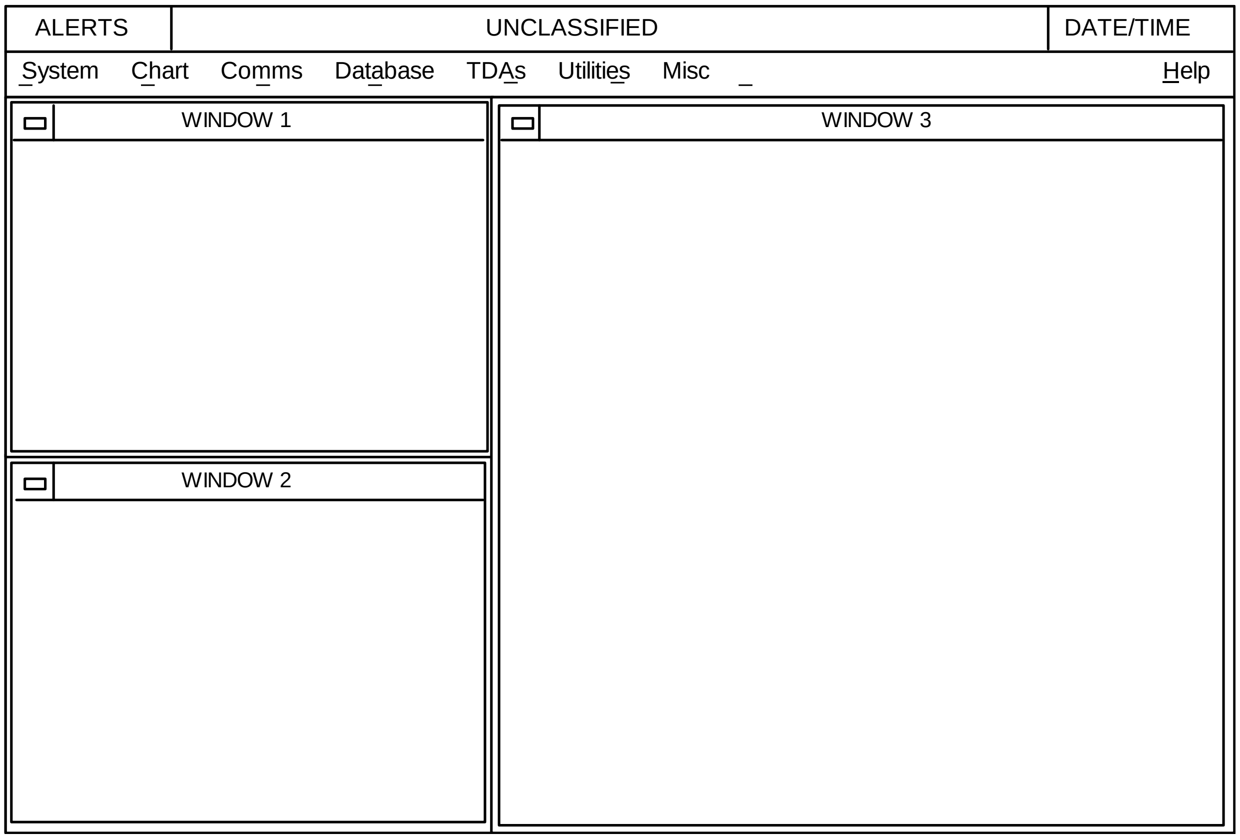


Figure 4-4. Example of tiled window placement (from the JMCIS style guide).

The specifications presented here call for an overlapping arrangement to be the default window placement. However, users should have the ability to arrange windows as desired on the screen and then to save and retrieve specific configurations as a user-preference setting. In addition, systems with operational requirements that dictate certain windows (e.g., the tactical map) not be obscured from user view should be able to define a tiled arrangement for these windows and to specify the placement of other transitory windows when they are displayed by users.

4.6 WINDOW ICONS

4.6.1 Appearance

A window icon provides a visual representation of a minimized window or window family. A window icon shall consist of a graphic image and a label, as shown in figure 4-5. The label shall be located below the image and have the same title as the corresponding window. The title may be truncated as needed so that the width of the label is the same as the width of the icon image. The location cursor shall appear on a window icon when it has input focus, and the full title of the icon shall be displayed. When the icon loses input focus, the title may be truncated again as needed.

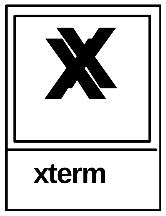


Figure 4-5. Example window icon (from the JMCIS style guide).

4.6.1.1 Placement

The default location for icons is the lower left corner of the screen, arrayed in rows from left to right, bottom to top.[24] An icon is placed in the position it last occupied if no icon is already there. If that position is taken, the icon is placed in the next free location. An option to change the default location where icons are displayed should be available as a user-selectable

[24]. Although the Motif Style Guide provides an icon box as an optional storage location for icons, developers should not implement an icon box in their applications.

preference setting. If desired, icons can be moved using the same methods available for moving windows.

4.6.1.2 Design of Window Icons

Developers should design unique graphic images for the window icons within the application so that users can easily recognize the application and/or function associated with an icon without reference to the icon label.[25] The default size of the icon image should be 50 x 50 pixels; maximum size should not exceed 64 x 64 pixels. Whenever possible, developers should select a concrete, rather than abstract, image to represent the application or function. For example, a map-related application might design the graphic for its icons to contain a miniature or stylized map.

The graphic image should be unambiguous (i.e., only one meaning can be attributed to it) and easily distinguished from other icons. Alternatively, if there are several classes of related objects within the application, the icons for the objects within a class should have the same shape but vary in their details. The graphic should provide a simplified representation since too much detail inhibits, rather than facilitates, perception. Developers should provide consistency in the shapes of families of icons and in identical icons of differing size by limiting variations in angles, line thicknesses, shapes, and amount of empty space. Developers should consider changing the appearance of an icon to indicate when a routine message (e.g., new mail has been received) has been generated by a process running in the minimized window. The graphic images used in an application should be developed in coordination with the system(s) in which the application will be available to ensure consistency with icons developed by other applications with similar functionality.[26]

The graphic image should be monochromatic, with a color that is similar to the colors used for other text and graphics in the system; this color should be used consistently in all window icons throughout the application. Dithering techniques should be used if a three-dimensional effect is desired. If color is used for coding purposes, it should be used redundantly. Developers should limit their use of color in icon design; the DoD style guide indicates that color provides no advantage in speed of recognition over representations in black and white and may negatively impact user performance by creating unnecessary display clutter. The use of text as part of the graphic should be minimized, given that the icon includes a label with the name of the window that has been minimized.

4.6.2 Behavior

[25]. Additional guidance on icon design can be found in Visual Design with OSF/Motif by Kobara and in the DoD style guide.

[26]. A set of standard icon graphics, including color values, will be included in a future version of this document.[27]. The inclusion of unavailable options in a menu is consistent with the Motif Style Guide but not with MIL-STD-1472D.

An icon has an Icon Menu containing the same options (except Size) as the Window Menu of the corresponding primary window. Minimize can be included in an Icon Menu but should not be available for selection. To display an Icon Menu, users shall place the pointer on the icon and click the Select button on the pointing device. Users shall select options in the menu using standard menu selection methods. The menu is dismissed when users click the Select button on the pointing device anywhere outside the menu. To restore the window family for a window icon, users shall place the pointer on the icon and double click the Select button on the pointing device, or select the Restore option from the Icon Menu.

5.0 MENUS

5.1 PULL-DOWN MENUS

5.1.1 Appearance

A pull-down menu, shown in figure 5-1, consists of a menu title and a set of options which are listed below the title and separated from it by a separator line. The title and each option in the menu should contain a mnemonic; an option may also include a keyboard accelerator as appropriate. The menu should be wide enough to accommodate the wording of the longest menu option plus a keyboard accelerator, and should provide enough space between them so that each can be easily read by users. The title of each pull-down menu shall be displayed in a menu bar along the top margin of a window.

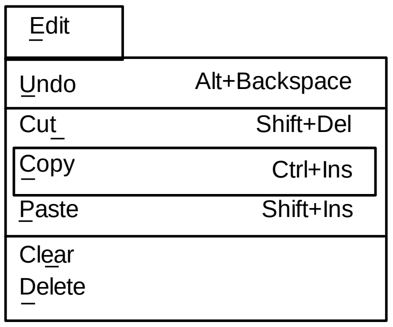


Figure 5-1. Example pull-down menu (from the JMCIS style guide).

5.1.1.1 Menu Title

The title of a menu should be a single word whenever possible. The title should describe the category or type of options presented in the menu and should be different from the other titles that appear in the menu bar. The first letter of each word in the menu title should be capitalized. If the title contains an acronym, it should be capitalized. The title should not contain an ellipsis or a right-pointing arrow.

5.1.1.2 Types of Menu Options

Menu options shall be one of three types, shown in figure 5-2: (1) actions that are executed as soon as they are selected, (2) routings that display a window or a cascading submenu, and (3) settings that are used to define parameters or specify an application state.

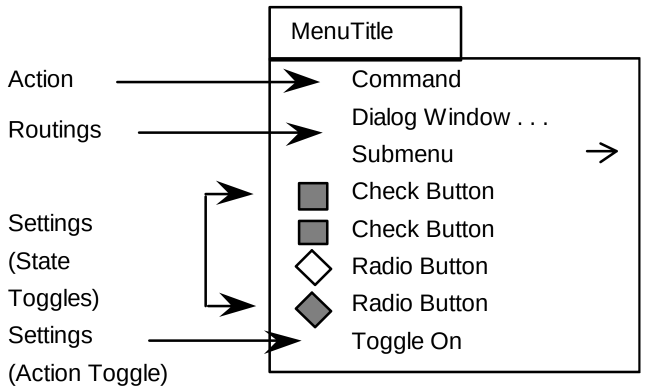


Figure 5-2. Examples of menu option types (from the JMCIS style guide).

Routing options that display a window shall be identified in a menu by an ellipsis (i.e., they are followed by ". . ."). Routing options that display a cascading submenu shall be followed by a right-pointing arrow.

Options that are settings may be action toggles (e.g., turn on/turn off) or state toggles (e.g., select a font size from the set available). Options that are action toggles should be worded to reflect the action that is implemented when the option is selected. The wording should be semantically congruent with natural usage; for example, if one toggle is worded "Move object up," the other toggle should be "Move object down," not "Move object back." When users select one of these options (e.g., turn on), the wording of the option in the menu should change (e.g., turn off) to reflect the action that will be implemented when the action is selected again. Only one of the action toggles shall appear in the menu at any time. If an "undo" option is provided, the wording should change dynamically to reflect the action that can be undone. For example, if the most recently executed action is cut, the option should be worded "Undo Cut."

Options that are state toggles should be worded to describe the state (e.g., a list of font names) and may be indicated by radio buttons (for exclusive settings) or check buttons (for nonexclusive settings), as appropriate, to the left of the option. When users select one of these options (by clicking on the menu option or the button), the check button or radio button should highlight but the option wording should not change.

5.1.1.3 Wording

Each menu option should be phrased to reflect the action that is executed by the option (i.e., phrased as a command to the computer rather than as a question to the user), and worded in the vocabulary of users rather than that of application developers. The vocabulary listed in table 6-1 should be used if the actions listed in the table are included in the wording of menu options.

Each option should be tersely worded (preferably limited to a single word) and presented in upper and lower case letters, with the first letter of each word capitalized. All acronyms should be capitalized. Each option should be left-justified and appear on a single line; long menu options should be accommodated by making the menu wider rather than making the item take two (or more) lines.

The wording of each menu option should be consistent in grammatical style and matched with the corresponding menu title. For example, additions to an Edit menu containing actions such as Cut, Copy, Paste, and Undo should be other verbs describing actions, rather than nouns describing objects or entities. An application should use verbs as the first word in a menu option because this format provides consistency and makes the menu options easier to read and understand. If desired, an application can provide supplementary information about menu options in the message area at the bottom of the window containing the menu; as the pointer is placed on an option, a more complete prose description of the action executed by the option is displayed.

5.1.1.4 Organization and Grouping

Developers that use pull-down menus in their applications should organize the options into logical or functional groups (as shown in figure 5-3), combine the groups into menus reflecting the categories of options addressed, and select titles that clearly describe each category. If the options cannot be organized into logical groups, they should be ordered according to frequency of usage within each menu, with the most frequently executed options at the top of the menu. Less frequently executed options and destructive options such as Delete and Clear should be at the bottom of the menu. Menu options that perform opposing actions (e.g., Save and Delete) should not be placed adjacent to each other in order to reduce the likelihood of accidental selection of an incorrect action. If similar options are included in different menus, the options should be ordered in a consistent manner in each menu. If an organization based on logical groups or frequency of use is inappropriate, then the options may be ordered alphabetically or in numerical order.

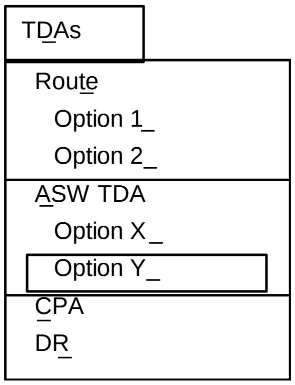


Figure 5-3. Example of logical ordering of menu options (from the JMCIS style guide).

An application menu should contain no more than 15 options. Menus with less than three options should be avoided. When a menu contains more than four options, the options should be divided into logical groups of no more than four options, and a separator line should be placed between the groups.

Because expert users benefit from positional constancy (i.e., having an option that is included in more than one menu presented in the same ordinal position in each of these menus), developers may choose to alter a menu organization based on logical groups in order to provide this form of consistency across menus in an application.

5.1.1.5 Available and Unavailable Options

If a menu option or or set of options is never available to a user (e.g., system administrator commands), the option(s) should not be included in a menu. If an option is only temporarily unavailable, it should be displayed in the menu but dimmed (as in figure 5-4) to indicate that it cannot be selected. For example, an option may be unavailable because it cannot be executed in the window that currently has input focus. The wording of options may change (e.g., when an option is an action toggle). However, options should not be added to or deleted from a menu to indicate their availability within a particular part of an application.[27]

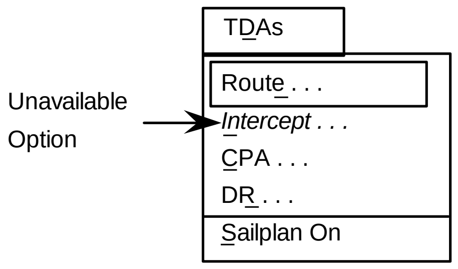


Figure 5-4. Example of an unavailable menu option (from the JMCIS style guide).

5.1.1.6 Cascading Submenus

Cascading submenus (see figure 5-5) should be used as needed to present groups of related actions hierarchically and shorten menus that are overly long (exceeding 15 options). The submenu shall be positioned either to the right of the parent menu or below the parent menu if space to the right is limited. The options in a submenu may themselves be the parent menu for another cascading submenu. The menu option associated with a cascading submenu should always be displayed as available. In addition, when this option is selected, the cascading submenu should always be displayed, even if all of the options in the submenu are unavailable.

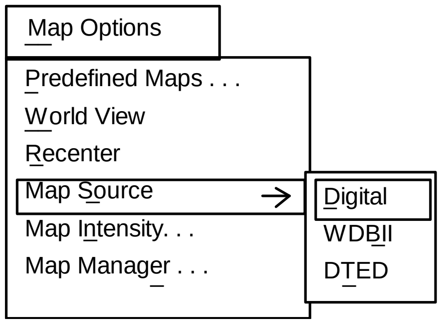


Figure 5-5. Example cascading submenu (from the JMCIS style guide).

A cascading submenu should contain only the options in that menu and should not repeat the parent option as the first option in the submenu. The submenu should be positioned so that its first option is aligned with the right-pointing arrow in the parent option for the submenu.

Applications should avoid using an excessive number of submenus. Submenus should be limited to three levels and organized as shown in figure 5-6. If the number of multiple submenus becomes too large, new menus should be created or a dialog window should be used.

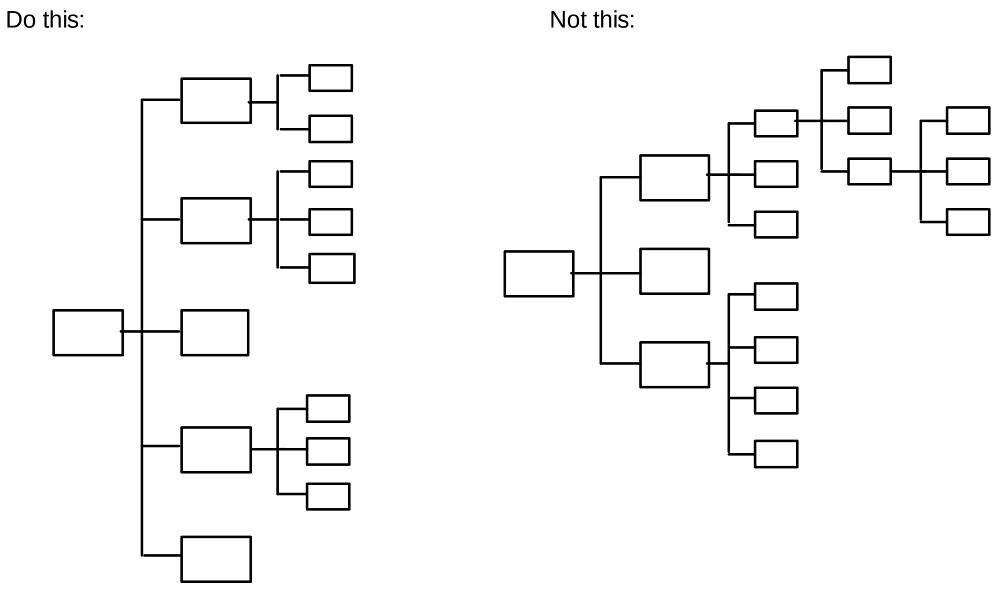


Figure 5-6. Organization of options in cascading submenus (from the DoD style guide).

5.1.2 Behavior

5.1.2.1 Pointing Device Navigation and Selection

Two methods, spring-loaded and posted, shall be available to activate (i.e., display) a pull-down menu and select a menu option.[28] In the spring-loaded method, users place the pointer on the menu title and press the Select button on the pointing device. The menu is displayed, with the location cursor on the first available option in the menu. As users drag the pointer over the menu options, the location cursor tracks the pointer, moving to each option as the pointer is placed on it, and any cascading submenus for which an option is the parent are displayed. The submenu is dismissed when the pointer moves off the parent option. Users drag the pointer to the option desired and release the Select button on the pointing device to select the option. The menu is dismissed, and the option is executed. Users drag the pointer off the menu and release the Select button to dismiss the menu without making a selection.

In the posted method, users place the pointer on the menu title and click the Select button on the pointing device to display the menu. The location cursor appears on the first available option in the menu. To display a cascading submenu, users place the pointer on the option that is the parent for the submenu and click the Select button on the pointing device. Users place the pointer on the option desired and click the Select button on the pointing device to select it. The location cursor moves to the option, the menu is dismissed, and the option is executed. Users move the pointer off the menu and click the Select button to dismiss the menu without making a selection.

5.1.2.2 Keyboard Navigation and Selection

Users shall press <F10> to move the location cursor to the first (i.e., leftmost) available menu title in a menu bar. If the keyboard has a <Menu> key, <Shift><Menu> shall have the same effect. If none of the menu titles is available, then <F10> (and <Shift><Menu>) shall not move the location cursor to the menu bar from its position in the window. Users shall press <Left> and <Right> to move the location cursor between available menu titles in the menu bar, with the direction of cursor movement wrapping between the last and first titles. <F10> (and <Shift><Menu>) shall exit a menu bar, and the location cursor shall return to the object that had input focus before the menu was activated.

To display a pull-down menu, users shall place the location cursor on the menu title and press <Down>.[29] When the menu is displayed, the location cursor appears on the first available option in the menu. Users shall press the arrow keys to move the location cursor between available options in the menu. The direction of cursor movement shall wrap from the bottom to top options in the menu. If the menu option with the location cursor is the parent for a

[28]. This document does not call for the assignment of default options in pull-down or pop-up menus although the Motif Style Guide supports this feature as optional.

[29]. The Motif Style Guide indicates that <Down> displays the pull-down menu while the Motif User's Guide indicates that <Return>, <Select>, or <Space> performs this action.

submenu, users shall press <Right> to display the menu and move the location cursor to the first available option in the submenu; <Up> and <Down> shall move the location cursor between options in the submenu, and <Cancel> shall dismiss the submenu and return the location cursor to the parent option for the submenu.

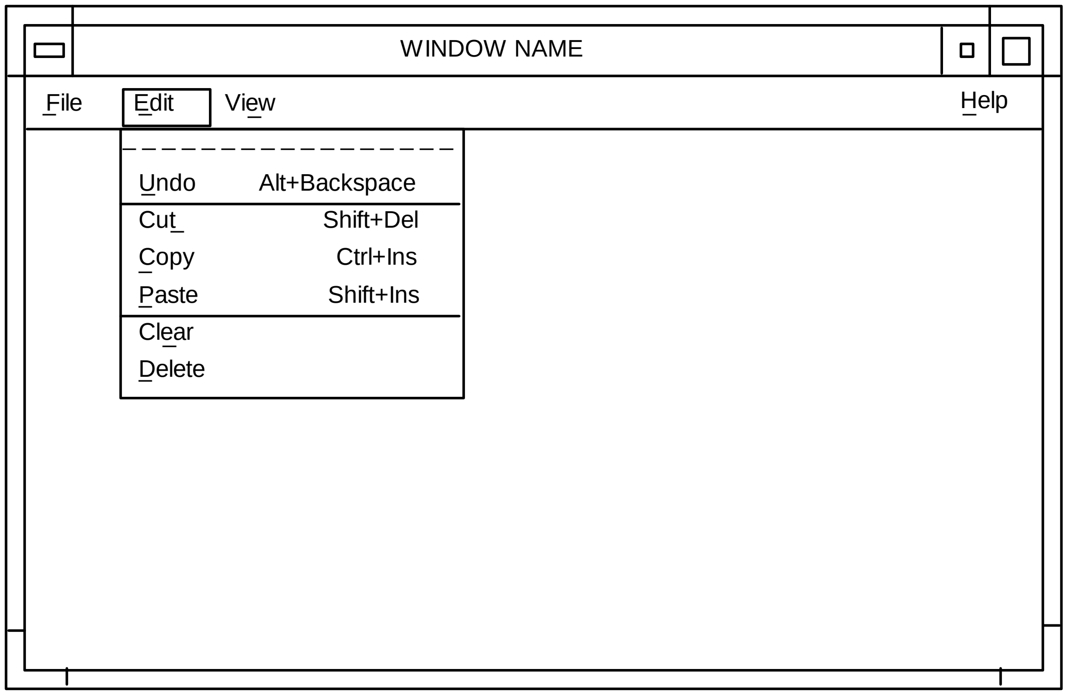
<Return>, <Enter>, or <Space> shall select an option in a pull-down menu and dismiss the menu. If the keyboard has a <Select> key, it shall also select an option and dismiss the menu.

5.2 TEAR-OFF MENUS

5.2.1 Appearance

A pull-down menu should provide a tear-off capability if users need to make multiple selections from the menu. A menu that can be torn off contains a dashed line graphic below the menu title and is the first selectable option in the menu, as shown in the top part of figure 5-7.[30] When users tear off a pull-down menu, the options in the menu (without the tear-off graphic) are displayed in a menu window, as shown in the bottom part of figure 5-7. The title of the window is the title of the pull-down menu. The contents of the menu window should be the same as the original menu, with the options presented in the same order in both cases. The availability of options in the menu window should be managed in the same way as in the original menu. For example, when the options in the menu window are unavailable for selection, they should be dimmed. In addition, the wording of options may change as selections are made in the menu window.

[30]. Tear-off functionality is controlled by the XmNtearOffModel resource of the RowColumn widget.



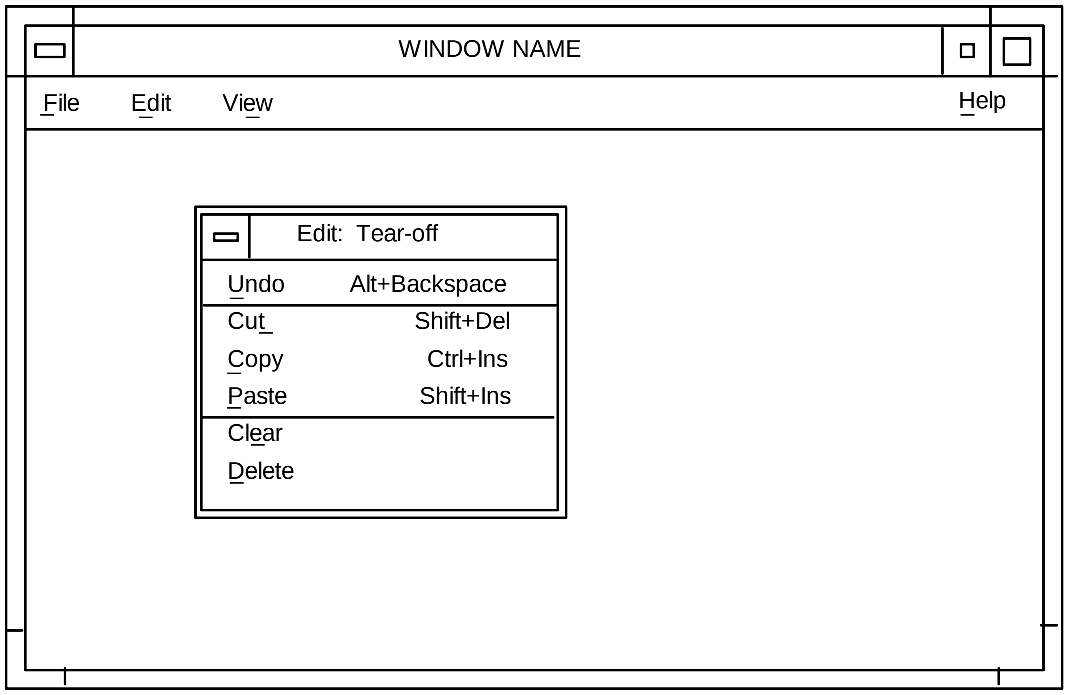


Figure 5-7. Example tear-off menu and menu window (from the JMCIS style guide).

5.2.2 Behavior

5.2.2.1 Pointing Device Navigation and Selection

To display a tear-off menu,[31] users shall display the associated pull-down menu, place the pointer on the tear-off graphic in the menu, and click the Select button on the pointing device. The pull-down menu is dismissed, its contents are displayed in a dialog window which is placed at the location of the pull-down menu. The window receives input focus, with the location cursor on the first available menu option.

To display a tear-off menu and move it to a new location, users shall display the associated pull-down menu, place the pointer on the tear-off graphic, and drag it with the Transfer button on the pointing device. When the Transfer button is released, the pull-down menu is dismissed, its contents are displayed in a dialog window which is placed at the current pointer position, and the window receives input focus, with the location cursor on the first available menu option.

To select an option in the menu window, users shall place the pointer on the option desired and click the Select button on the pointing device. The location cursor moves to the option, the option is executed, and the window remains displayed so that the menu is available for additional selections. Users shall dismiss the window by selecting Close in the Window Menu or pressing <Cancel> when the window has input focus.

Users should be able to manipulate (e.g., move, close) the menu window in the same ways as other dialog windows (see section 4.3). Users should be able to display the pull-down menu associated with a menu window while the window is displayed and select options either from the original pull-down menu or from the options in the menu window. If the tear-off option in the menu is selected, the original window should be dismissed and replaced with a new version of the menu window.

5.2.2.2 Keyboard Navigation and Selection

To display a tear-off menu, users shall activate the associated pull-down menu as indicated in section 5.1.2.2 and press the arrow keys to move the location cursor to the tear-off graphic in the menu. <Return>, <Enter>, or <Space> (and <Select> if available) shall dismiss the pull-down menu, display its contents in a menu window which is placed at the location of the pull-down menu, and assign input focus to the window, with the location cursor on the first available menu option.

Users shall press the arrow keys to move the location cursor between menu options. <Return>, <Enter>, or <Space> (or <Select> if available) shall select an option. The

[31]. Tear-off menus are a new capability that was added to Motif 1.2. Applications using earlier versions of Motif are not required to implement this menu feature.

menu window shall remain displayed until dismissed, either by selecting Close in the Window Menu or pressing <Cancel>. Users can manipulate (e.g., move, close) the menu window from the keyboard in the same ways as other dialog windows (see section 4.3).

5.3 POP-UP MENUS

5.3.1 Appearance

A pop-up menu is hidden from view and is displayed only when selected with the pointing device or keyboard. If a pop-up menu includes a title, it should be displayed at the top of the menu, separated from the first menu option by a separator line. A pop-up menu should contain no more than 15 items. Although pop-up menus can include cascading submenus, their use is not recommended. Pop-up menus may include mnemonics but should not contain keyboard accelerators. In addition, a pop-up menu may include options that are temporarily unavailable (indicated by dimming, as in pull-down menus).

The contents of a pop-up menu depend on the element with which it is associated in a window. For example, the pop-up menu for a text field should contain edit-related options, whereas the pop-up menu that is displayed when the pointer is on the background area of a window should contain options that pertain to global operations within the window. When the pointing device is used, the menu contents relate to the element on which the pointer is placed. When the keyboard is used, the menu contents relate to the element with the location cursor. In both cases, the pop-up menu should be displayed near the element with which it is associated.

A window containing a pop-up menu should provide some indication that the menu is available (e.g., include text in the message bar indicating the presence of the menu in the window). In addition, if pop-up menus are included in an application, they should provide a redundant method for accessing frequently performed actions in a window and should not provide access to new actions. For example, an application where users access editing commands from a pull-down menu might also provide a pop-up menu for the text fields in the window in order to reduce the amount of pointer movement required to perform an editing operation.

If a pop-up menu contains the same options as a pull-down menu, the order of the options should be the same in the two menus; similarly, if a pop-up menu contains options that have keyboard accelerators assigned to them in the pull-down menus, then the same accelerator should be available in the pop-up menu and have the same effect when executed.

5.3.2 Behavior

5.3.2.1 Pointing Device Navigation and Selection

Users shall use the spring-loaded and posted methods described in section 5.1.2.1 to display a pop-up menu, navigate in the menu, and select a menu option. In the case of pop-up menus, however, a menu is displayed when users press or click the Menu button on the pointing device. In addition, users shall be able to navigate within a pop-up menu and select an option with either the Select or Menu button on the pointing device. When a pop-up menu is displayed, the location cursor shall appear on the first available option in the menu.

5.3.2.2 Keyboard Navigation and Selection

To display a pop-up menu, users shall move the location cursor to a window area where a pop-up menu is available and press <Shift><F10>. If the keyboard has a <Menu> key, it shall also activate the pop-up menu. When the menu is displayed, the location cursor is on the first available option in the menu. As with pull-down menus, users shall press the arrow keys to move the location cursor between options in a pop-up menu.

<Return>, <Enter>, or <Space> shall select an option in a pop-up menu and dismiss the menu. If the keyboard has a <Select> key, it shall also select an option and dismiss the menu. <Cancel> and <Shift><F10> (and <Menu> if available) shall dismiss a pop-up menu. In each case, the location cursor shall return to the object that had input focus before the menu was displayed.

5.4 OPTION MENUS

5.4.1 Appearance

An option menu, such as the one shown in figure 5-8, consists of a title and up to 15 options, only one of which is visible in the option menu button. The title shall be a label (rather than a push button, as is the title of a pull-down menu) that appears next to or above the option menu button. The option displayed shall be the last selection made by users. The option should contain a mnemonic and be followed by a bar graphic to distinguish the option menu button from an action push button. The option menu button should be large enough to accommodate the longest menu option and the bar graphic; the text for an option should not be obscured by the bar graphic.

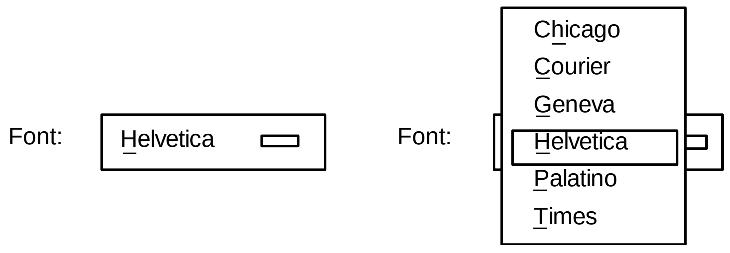


Figure 5-8. Example option menu (from the JMCIS style guide).

5.4.2 Behavior

5.4.2.1 Pointing Device Navigation and Selection

Users shall use the spring-loaded and posted methods described in section 5.1.2.1 to display an option menu, navigate in the menu, and select a menu option. When an option menu is displayed, the location cursor shall appear on the previously selected option. When an option is selected, it shall be displayed as the label in the option button.

5.4.2.2 Keyboard Navigation and Selection

To display an option menu, users shall move the location cursor to the option button for the menu and press <Space>. If the keyboard has a <Select> key, it shall also activate the option menu. When the menu is displayed, the location cursor is on the first available option in the menu. As with pull-down menus, users shall press the arrow keys to move the location cursor between options in an option menu.

<Return>, <Enter>, or <Space> shall select an option in an option menu and dismiss the menu. If the keyboard has a <Select> key, it shall also select an option and dismiss the menu. In both cases, the option selected shall be displayed as the label in the option button. <Cancel> shall dismiss an option menu, and the location cursor shall return to the object that had input focus before the menu was displayed.

5.5 MNEMONICS

5.5.1 Appearance

Applications should assign single-character mnemonics, such as those in figure 5-9, to menu titles in a menu bar and to menu options in pull-down and pop-up menus so that an additional method of menu selection from the keyboard is available to users. The mnemonic for each title in a menu bar and for each option in a menu must be unique, but the same character can be used as a mnemonic in different menus. Whenever the same menu title or option appears in an application, it should have the same mnemonic.

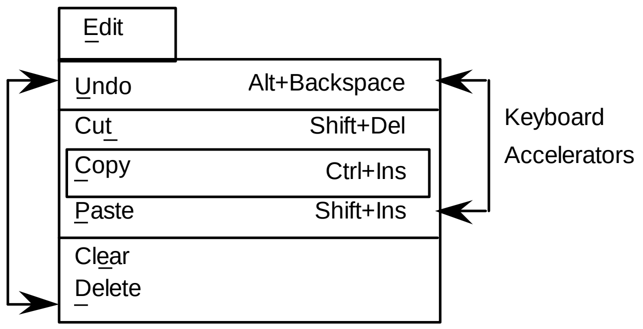


Figure 5-9. Example mnemonics & keyboard accelerators (from the JMCIS style guide).

Application developers should use the mnemonics listed in table 5-1 in menu titles and options. When selecting the mnemonic for a menu title or option, developers should try to select characters to produce minimal interference with the mnemonics used in other menus. For example, the same character should not be assigned as the mnemonic for options performing opposite or contradictory actions in different menus (e.g., <C> used as the mnemonic for Continue in one menu and Close in another). In addition, developers should assign mnemonics to be coordinate with the key combinations selected as keyboard accelerators. For example, <S> might be the mnemonic and <Ctrl><S> the keyboard accelerator for a menu option to save a file.

Table 5-1. Standard mnemonics and keyboard accelerators.[32]

Action	Mnemonic	Accelerator
Clear	E	-----
Close	C	<Alt><F4>
Context-Sensitive Help	C	<Shift><Help>
Copy	C	<Ctrl><Insert> or <Copy> or <Ctrl><C>
Copy Link	K	-----
Cut	T	<Shift><Delete> or <Cut> or <Ctrl><X>
Delete	D	<Delete>
Deselect All	L	<Ctrl><\>
Edit	E	-----
Exit	X	-----
File	F	-----
Find	F	-----
Help	H	<Help> or <Shift><F1> or <Shift><Help>
Help Index	I	-----
Include	I	-----
Index	I	-----
Keyboard	K	-----
Lower	L	<Alt><F3>
Maximize	X	<Alt><F10>
Minimize	N	<Alt><F9>
Move	M	<Alt><F7>
New	N	-----
Open	O	-----
Overview	O	-----
Paste	P	<Shift><Insert> or <Paste> or <Ctrl><V>
Paste Link	L	-----
Primary Copy	---	<Alt><Ctrl><Insert> or <Alt><Copy> or <Alt><Ctrl><C>
Primary Move	---	<Alt><Shift><Delete> or <Alt><Cut> or <Alt><Ctrl><X>
Print	P	-----
Redo	R	<Shift><Alt><Backspace>
Refresh	E	<F5>
Reselect All	---	<Alt><Insert>
Restore	R	<Alt><F5>
Save	S	-----
Save As	A	-----
Select All	S	<Ctrl></>
Size	S	<Alt><F8>
Sort	S	-----
Tutorial	T	-----
Undo	U	<Alt><Backspace> or <Undo> or <Ctrl><Z>
Using Help	U	_____

[32]. These mnemonics and accelerators are taken from the Motif Style Guide, and the accelerators are consistent with (i.e., duplicate) the key bindings listed in table 2-2. The set of standard accelerators to be used by applications will be expanded in subsequent versions of this document.[33]. The actions performed by GoBack, Restart, Review, and Suspend are taken from the DoD style guide.

The character assigned as the mnemonic shall be underlined. Whenever possible, the mnemonic should be the first letter of the label for a menu title or option. Another letter in the label can be selected if the same first letter occurs in more than one title in the menu bar or in more than one option in the menu. If necessary, a character that is not part of the label can be selected to prevent duplicate characters from being selected. In addition, the options in a menu can be numbered sequentially, and the number can be used as the mnemonic. If a character that does not appear in the label is assigned as a mnemonic, it shall appear in parentheses following the label.

5.5.2 Behavior

To move the location cursor to the menu bar in the window with input focus, users press <Alt> and type the mnemonic for an available title in the menu bar. The pull-down menu associated with the title shall be displayed, with the location cursor on the first available option. If the location cursor is already on a title in the menu bar, users shall type only the mnemonic to display the menu. Users then type the mnemonic for an option in the menu to select the option and dismiss the menu. Mnemonics shall not be case sensitive; users shall be able to type the character in either upper or lower case.

To use mnemonics in a pop-up menu, users shall display the menu as indicated in section 5.3.2.2, then type the mnemonic for an option to select it and dismiss the menu. In an option menu, users press <Alt> and type the mnemonic to display the menu, then type the mnemonic for an option to select it and dismiss the menu.

5.6 KEYBOARD ACCELERATORS

5.6.1 Appearance

Applications should implement keyboard accelerators, shown in figure 5-9, to provide keyboard access to frequently executed menu options. Keyboard accelerators can be provided for action options that are commands, for routing options that display a window, and for options that are settings. If an accelerator is available for a menu option, it should be right justified on the same line as the option and separated by enough space to make it visually distinct.

Keyboard accelerators should include a plus sign to indicate the combination of keys (e.g., Shift+Ins) that must be pressed at the same time. Also, the keys should be identified by their engravings on the keyboard; for example, if the <Control> key is labeled Ctrl, any accelerator using this key should read Ctrl, not Control. The keyboard accelerators created by an application should be of the form “modifier + character,” where the modifier is <Alt>, <Ctrl>, <Shift>, or a combination of these keys, and the character is an alphanumeric or special key on the keyboard. The accelerator for an option that has been assigned a mnemonic should be “modifier + mnemonic,” whenever possible.

Application developers should use the key combinations listed in table 5-1 when assigning keyboard accelerators to these actions. The same key combination should be assigned to a keyboard accelerator whenever it is used in an application and, if possible, across applications within a system. Developers can define additional accelerators as needed but should ensure that the key combinations selected are not already bound to other actions (see table 2-2, appendix D, and the sections on keyboard navigation and selection). Developers should not define accelerators that are a combination of <Alt> and letter keys (which might conflict with mnemonics) or a combination of <Shift> and letter keys (which might conflict with keystrokes used in text entry).

Because a system is likely to be installed on multiple hardware platforms with keyboards that require different mappings to Motif virtual keys (see table 2-3), users are likely to execute a keyboard accelerator using different key combinations depending on platform at which they are working. To minimize possible confusion, each system should ensure that the keyboard accelerator listed for each menu option matches the particular keyboard being used. In addition, in cases where multiple accelerators are available for executing a single action (e.g., for edit commands such as Cut, Copy, and Paste), each application should support all of the accelerators (even if only one can be listed with the menu option) so that users can continue to execute these actions as they move between different platforms. Finally, the specific keys that execute the keyboard accelerators on each keyboard should be included in the system-level Help menu by each system (see section 9.1)

5.6.2 Behavior

To select a menu option using a keyboard accelerator, users shall press the keys associated with the option. The menu containing the option shall be displayed briefly to provide a visible response to the user action, then the option is executed in the window with input focus. If a keyboard accelerator contains an alphabetic character, the accelerator should not be case sensitive; users should be able to type the character in either upper or lower case.

6.0 CONTROLS

6.1 PUSH BUTTONS

6.1.1 Appearance

A push button shall be used to initiate an action. A push button is a rectangular object containing either a short label describing the button's action or an icon representing the action that the button performs. Figure 6-1 presents examples of push buttons containing labels and action icons.

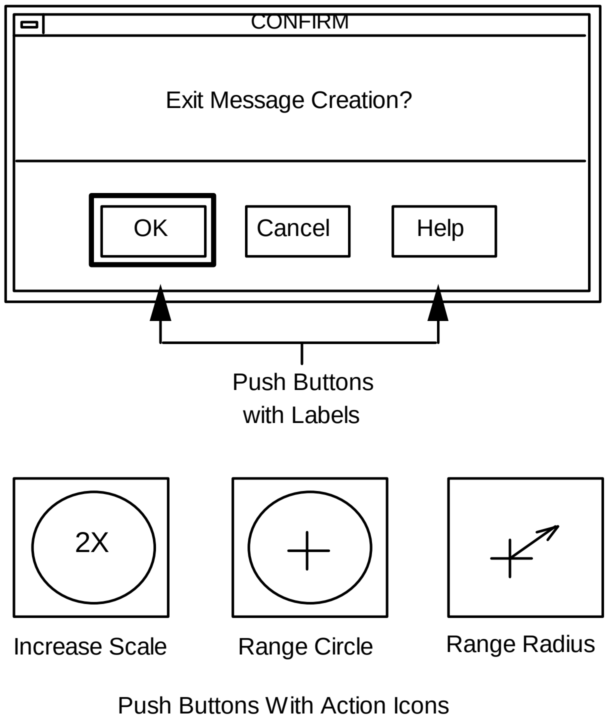


Figure 6-1. Example push buttons and action icons (from the JMCIS style guide).

The label in a push button should be mixed case, with the first letter of each word capitalized. Push button labels (whether text or graphics) should be large enough to be easily read and interpreted by users at normal viewing distances (see section 8.4.1 on text font, size, and legibility). In addition, there should be enough space between the label and the rectangle surrounding it not to restrict the legibility or visibility of the text or graphic in the push button. Kobara, in Visual Design with OSF/Motif, recommends a margin height of 2 pixels and a margin width of 8 pixels for push buttons. The push button should contain an ellipsis if selecting the push button results in another window (other than a help window or a message window to confirm the action executed by the push button) being displayed.

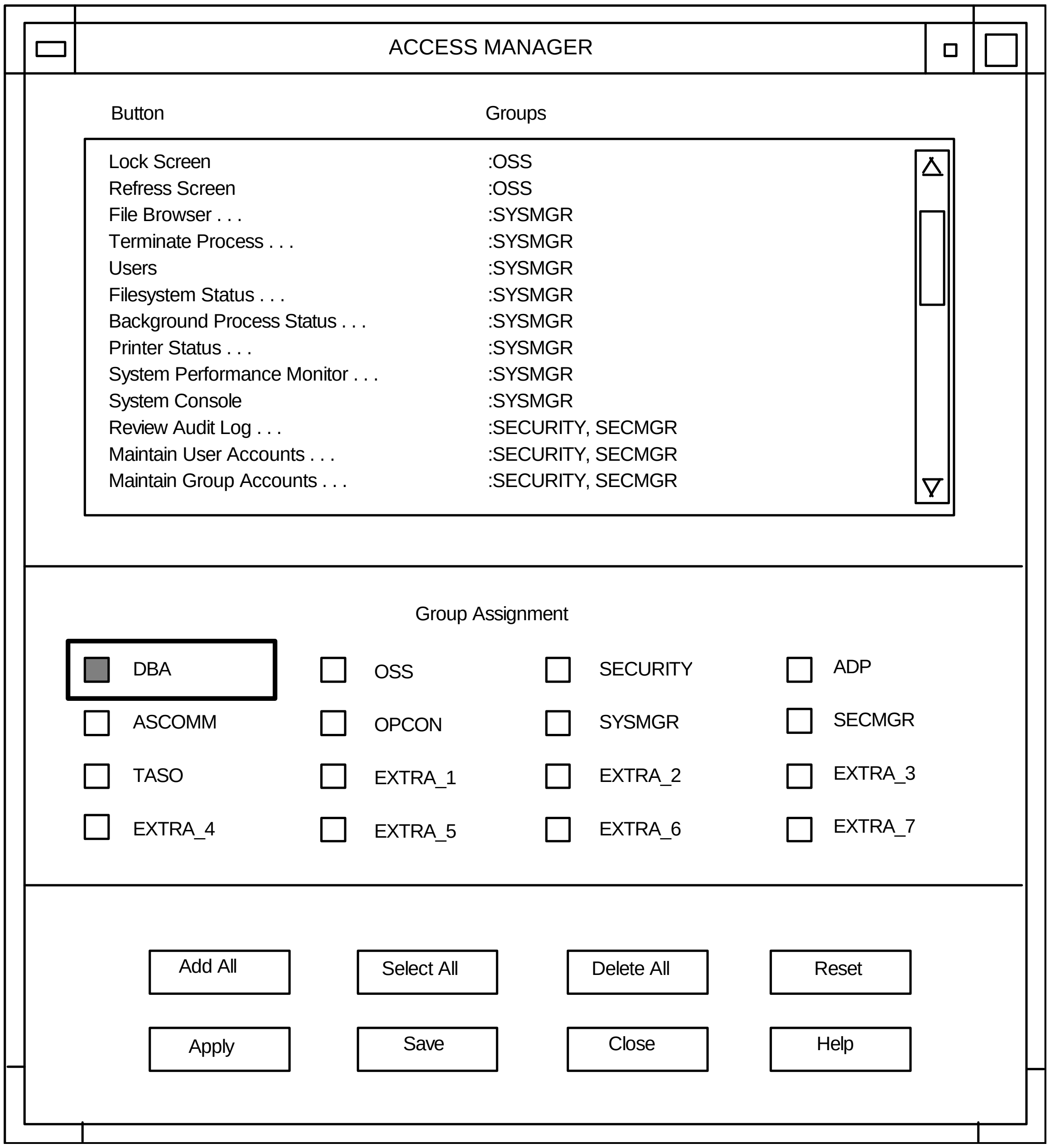
Push buttons should be the same size within a window. If the push buttons contain text labels, the buttons should be wide enough to display the longest button label. If the push buttons contain icons, the buttons should be large enough to display the largest icon. Exceptions to these size guidelines may occur in order to accommodate a push button with a label or graphic that is significantly longer or larger than the others in the window, especially when space within a window is limited.

6.1.1.1 Standard Vocabulary

Push button labels should be short and unambiguous. For action buttons, the label should describe the results of pressing the button and reflect the action that will be taken by the application rather than the user. For example, “hook a track” is not an appropriate label for a push button because it indicates the action that is executed by the user; this information is better presented as a message in the message bar of the window. When OK or Yes are used as response buttons, the question or statement that prompts the response should be worded carefully (e.g., phrased without negatives) and be unambiguous.

Each push button in a window should represent a unique action that users can perform in a window. Developers should identify the set of actions that users can perform in the application, select a single term to describe each action, and then use that term whenever users need to perform the action. An application window should not contain multiple push buttons that appear to perform the same action.

Terms such as “All” should be used in push button labels (e.g., Select All, Delete All) only when there is no ambiguity as to the object or data element to which “All” refers. If a push button label may have multiple referents within a window, the name of the object or data element should be used in the label instead of “All.” For example, the access manager window shown in figure 6-2 contains two objects (i.e., a list of groups and a set of group assignments) and a number of push buttons, three of which include the term “All.” Because this window contains multiple objects, it is important that users know what the referent is for the actions performed by these buttons (i.e., which objects do they affect) and whether the referent is the same for each button (i.e., do the buttons affect the different objects). Confusability can be reduced by renaming the buttons to indicate the object to which they refer, or by redesigning the window to locate the push buttons near the object they affect.



ACCESS MANAGER

Button	Groups
Lock Screen	:OSS
Refress Screen	:OSS
File Browser . . .	:SYSMGR
Terminate Process . . .	:SYSMGR
Users	:SYSMGR
Filesystem Status . . .	:SYSMGR
Background Process Status . . .	:SYSMGR
Printer Status . . .	:SYSMGR
System Performance Monitor . . .	:SYSMGR
System Console	:SYSMGR
Review Audit Log . . .	:SECURITY, SECMGR
Maintain User Accounts . . .	:SECURITY, SECMGR
Maintain Group Accounts . . .	:SECURITY, SECMGR

Group Assignment

■ DBA	☐ OSS	☐ SECURITY	☐ ADP
☐ ASCOMM	☐ OPCON	☐ SYSMGR	☐ SECMGR
☐ TASO	☐ EXTRA_1	☐ EXTRA_2	☐ EXTRA_3
☐ EXTRA_4	☐ EXTRA_5	☐ EXTRA_6	☐ EXTRA_7

Figure 6-2. Example of push button terminology where the referent is ambiguous (from the JMCIS style guide).

Application developers should use the vocabulary listed in table 6-1 to construct push button labels whenever applications perform the actions indicated in the table. Developers can create new vocabulary as needed to describe actions not listed in the table. If a new action is created, it should be a verb, stated in active voice, and describe the action that pressing the button causes. The names of actions should be congruent (e.g., Save/Delete, On/Off, In/Out). Developers should not create new vocabulary for actions already defined in the table, or use existing vocabulary to describe new actions. In addition, developers should ensure that the same vocabulary is used to describe the same action throughout the application.

Table 6-1. Action vocabulary.

Term	Action Performed
MANIPULATING PRIMARY WINDOWS	
	These commands apply to windows where users can perform multiple actions (vice dialog windows where users usually perform a single action). This vocabulary can be used in both push buttons and menu options.
Back	Moves a window to the back of the screen.
Close	Closes the current primary window and its secondary windows in an application with multiple primary windows; requests confirmation if unsaved changes have been made and allows the user to save the changes.
Exit	Closes all primary and secondary windows in an application and ends processing by the application.
New	Opens a new window; supplies a default name for the new window when it is opened.
Open	Opens an already defined window; prompts the user for the name of the window if more than one can be opened.
Open As	Opens an existing window; prompts the user for changes to how the window is to be presented (e.g., different format).
Print	Initiates a process for printing the contents of a window (e.g., parts of a window, full window).
Refresh	Redraws the contents of a window; updates the contents of the window to reflect the current state of the underlying data.
Save	Saves the contents of a window to a storage device.
MANIPULATING FILES AND DIRECTORIES	
	These commands may be combined with the object of the command (e.g., Print File, Print Directory) if multiple commands and multiple objects are available in a window. This vocabulary can be used in both push buttons and menu options.
Archive	Creates a backup copy of a file (e.g., on magnetic tape or fixed disk).
Close	Closes a file; requests confirmation if unsaved changes have been made to the file and allows the user to save the changes.
Delete	Removes a file from a storage device; requests confirmation prior to deletion.
Duplicate	Creates a copy of a file and prompts the user to name the file.
New	Opens a new file; supplies a default name for the new file when it is opened.
Open	Opens an existing file; prompts the user for the name of the file if more than one can be opened.
Open As	Opens an existing file; prompts the user for changes to how the file is to be presented (e.g., different format).
Print	Initiates a process for printing the contents of a file.
Rename	Renames a file; prompts the user to name the file; does not affect the contents of the file.

Restore	Retrieves the backup copy of a file and saves it to a storage device.

Table 6-1. Action vocabulary (continued).

Term	Action Performed
Revert	Replaces the current file with the version that was most recently saved.
Save	Saves a file to a storage device under the same file name; prompts the user for a name if the file does not have one. Does not remove the existing contents from the screen. Saves the file under the current file name, and overwrites any existing file with the current file name without need for user verification. Prompts the user for overwrite verification if the current file still has the default file name and a file already exists with that name. Invokes the Save As function if the file has no current file name.
Save As	Saves a copy of a file under a new name; prompts the user for the new name. Does not remove the existing contents from the screen. Prompts the user for the new name and for overwrite verification if a file already exists with the same name as entered by the user.
EDITING TEXT OR GRAPHICS OBJECTS	
Clear	Removes an object from a window without copying it to the clipboard; does not compress the remaining space.
Copy	Duplicates an object in a window and copies the object to the clipboard.
Cut	Removes an object from a window and stores it in the clipboard.
Delete	Removes an object from a window without copying it to the clipboard; compresses the remaining space.
Find	Locates an item (e.g., item in a list, word in a text area) or object that matches criteria specified by the user.
Paste	Inserts an object from the clipboard into a window at a selected location.
Redo	Reverts an Undo; i.e., returns an object to its state after the last operation was performed.
Replace	Replaces a word or character string with a different word or character string entered by the user.
Spell Check	Compares the words in a file or text selection against a dictionary of recognized words and identifies entries not in the dictionary (e.g., misspelled words).
Undo	Returns an object to its state before the last operation was performed.
MANIPULATING ITEMS	
	"All" may be appended to these commands (e.g., Delete All) to indicate that the command applies to all items in a set of items.
Add	Adds a new item to a set of items.
Append	Adds new information to the end of an item, or adds a new item to the end of a set of items.
Compare	Displays information on multiple items in a set of items.
Delete	Removes an item from a set of items.
Describe	Displays a detailed explanation or description of an item.
Deselect	Deselects (and removes highlight from) all items in a set of items.

All	
Duplicate	Creates a copy of an item and adds it to a set of items.
Edit	Edits an item in a set of items.

Table 6-1. Action vocabulary (continued).

Term	Action Performed
Insert	Adds an item to a specific location within a set of items.
Mark	Annotates (e.g., with an asterisk) that an item has been selected.
Merge	Combines the contents of two items into a single item.
Next	Displays the contents of the next item in a set of items.
Reselect	Reselects (and highlights) all of the items in the most recently performed selection.
Select All	Selects (and highlights) all items in a set of items.
Select On	Allows the user to select a subset of items based on criteria specified by the user.
Sort	Arranges a set of items in an order specified by the user.
Transmit	Transmits an item via a communication channel selected by the user.
Unmark	Removes the annotation that an item has been selected.
View	Displays the contents of an item.
MANIPULATING MAPS	
Center On	Centers the map on a position chosen by the user (e.g., current pointer position, own ship, selected track, chosen site).
Center/ Width	Displays a new view of the map using center and width entered by the user.
Default	Returns the map to a default view defined by the system/user (e.g., Default Chart returns to the default chart display).
Double	Doubles the width of the current map.
Half	Halves the width of the current map.
Previous	Returns to the map prior to the current one (e.g., Previous Chart displays the chart viewed prior to the current one).
Zoom	Moves the view of the map inward or outward.
CONSTRUCTING AND EXECUTING QUERIES	
	These commands are used in database query commands where the user wants to retrieve a set of items based on specific criteria (e.g., data fields).
Browse	Allows the user to navigate through the database prior to executing a query.
Compile	Generates an executable version of the query and checks it for correctness.
Execute	Performs a function, procedure, or process associated with the window.
Modify	Allows the user to make changes to an existing query.
Save	Saves a query to a storage device, providing the query with a unique name that identifies

it as a query.

Select	Identifies the fields to include in the query.
Show	Allows the user the define the manner in which the results of the query are to be displayed.

Table 6-1. Action vocabulary (continued).

Term	Action Performed
PAGING	
First	Displays the first page of information.
Last	Displays the last page of information.
Next	Displays the next page of information.
Previous	Displays the previous page of information.
PROCESSING	
	"On" and "Off" may be appended to the name of a specific process to start and stop the process; for example, if "Monitor" is an automatic monitoring process, "Monitor On" starts the process and "Monitor Off" stops the process.
GoBack[33]	Displays the previous transaction.
Pause	Interrupts a process without changing data entries or control logic for the process.
Restart	Cancels entries made in a transaction sequence and returns to the beginning of a sequence.
Resume[34]	Resumes a process that was previously paused (e.g., turns on a process that has been turned off).
Retry	Executes a halted process again.
Review	Returns to the first display in a transaction sequence so that users can make changes if desired.
Start	Begins or turns on a process.
Stop	Ends or turns off a process at the next nondestructive breaking point.
Suspend	Preserves the current transaction status when users log out of the system and permits resumption of work when users later log in to the system.
Update	Checks the status of a process and displays the updated information.
EXECUTING CONTROL SETTINGS	
Apply	Executes the control settings in a window but does not close the window.
Cancel	Closes a window without executing the control settings in the window.

[34]. The definition of this action is taken from the Motif Style Guide. The DoD style guide recommends the use of Continue to resume a process without changing data entries or control logic for the process.

Table 6-1. Action vocabulary (continued).

Term	Action Performed
Close[35]	Closes a window without executing the control settings in the window; used only in a window when performing actions that are irreversible.
Defaults	Restores all values in a window to a default state defined by the application.
OK	Executes the control settings in a window and closes the window.
Reset[36]	Cancels any changes made to the control settings in a window that have not been applied by the application, and resets the window to the state at the last time a change was applied or to the window's initial state.
EXECUTING OTHER WINDOW ACTIONS	
Help	Displays on-line information about an item or general information about a window.
No	Indicates a negative response to a question and removes the window containing the question.
Yes	Indicates an affirmative response to a question and removes the window containing the question.
SYSTEM-LEVEL COMMANDS	
Logout	Ends processing by a system; closes all windows on the workspace and stops all processing; requests confirmation if unsaved changes have been made and allows the user to save the changes.

6.1.1.2 Default Push Buttons

Applications should identify a default action in each dialog window and designate a default push button for this action. The default push button shall be indicated by an extra border around it (as in figure 6-3). In general, one push button should be designated as the default in a

[35]. Cancel should be used in a window when the settings selected by users are reversible (i.e., not saved when the window is removed from the screen). The settings revert to their original state so that when the window is displayed again, the user sees these values, not the ones selected when the window was last closed. Close should be used in a window when the settings are saved, when a new state has been defined, or when new data have been generated as a result of selections made in the window. The action associated with Close is irreversible in that the information, state, or settings do not revert when the window is closed.

[36]. The DoD style guide recommends the use of Cancel to erase changes just made by the user and restore the display to its previous state.

window, and the same button should be the default whenever the window is displayed. When the action defined by a push button can be executed by pressing <Return> in a window, the push button shall have the appearance of a default push button. The default designation may assigned to a different push button depending on the object that has keyboard focus in the window.

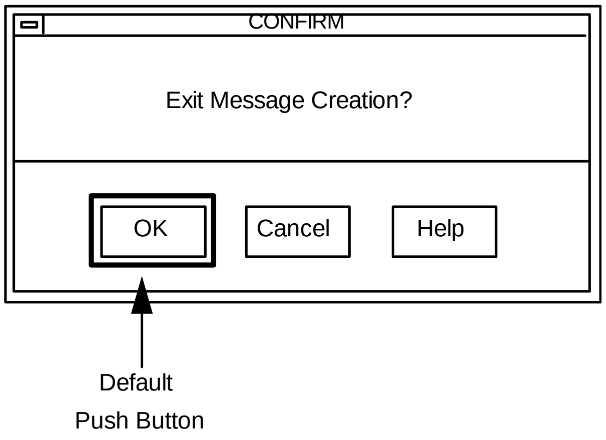


Figure 6-3. Example default push button (from the JMCIS style guide).

When the keyboard is used to navigate in a group of push buttons, the default designation should be mapped to the location cursor so that the default designation moves with the cursor. However, when keyboard focus is assigned to a different tab group, the default designation should return to the push button that was originally the default in the window.

OK is frequently the default push button in dialog windows. OK and Yes should not be the label for the default button if the action executed by the button is potentially destructive. In this case, the default should be Cancel so that the user is required to change the selection in order to perform the destructive action. If possible, applications should be designed so that the action performed by the default push button is reversible. If there is no default button in a window, <Return> should have no effect, and users should have to select one of the available push buttons to execute an action.

6.1.1.3 Design of Action Icons

The graphics for action icons should be designed so that users can easily identify the function performed when the icon is selected.[37] Each graphic should be unambiguous and easily distinguished from any other action icons with which it is displayed. Developers should consider including a short text label as part of the graphic if the function being represented is

[37]. A set of action icons for frequently executed actions will be included in a future version of this document.

highly abstract. The graphics for action icons that represent opposite functions (e.g., Save, Delete) should be designed to mirror each other. The graphics should be presented in a common style (e.g., the same line thickness to draw shapes) and oriented consistently when presented within the rectangular push button. A set of action icons should be presented in a push button palette and positioned within application windows following guidelines in section 8.1.3.4. An ellipsis is normally used to indicate that a push button or menu option, when selected, will result in a dialog window being displayed. In the case of action icons, an ellipsis should be omitted since the space available for text labels is extremely limited.

There are several different ways to create the icons for a set of actions such as those that are included in a palette. An icon may be designed by depicting a before and after representation of the action (e.g., a small and large version of an object, connected by an arrow, to indicate Magnify), by depicting the tool that would accomplish the action (e.g., a pair of scissors to indicate Cut), or by depicting the action itself (e.g., a paintbrush filling an object with color to indicate Paint). When designing the icons for a palette, developers should select one of these schemes and generate all of the icons to fit this scheme.

6.1.2 Behavior

The Select button on the pointing device shall be used to select a push button. To not execute the action, users shall move the pointer off the button while the Select button is pressed and then release the button. When the Select button is pressed, the push button shall highlight (i.e., change color) and change from raised to recessed appearance. When the Select button is released, the push button shall return to its normal appearance and the action represented by the push button shall be executed.

When the location cursor is on a push button, <Space> (and <Select> if available) shall select it from the keyboard. To execute the action represented by a default push button, users shall perform the operations described in Section 3.3.3.1.

6.2 RADIO BUTTONS

6.2.1 Appearance

A radio button shall be used to select one option from a group of options. Radio buttons are exclusive settings; they shall be used when selecting a single option from a set of at least two mutually exclusive options. A radio button consists of a diamond-shaped indicator, followed by a label describing the option represented by the button, as shown in figure 6-4. The label should define the state being set by the user. The text should be displayed in mixed case, with the first letter of the option capitalized. All of the radio buttons in a window should be the same size. If a radio button is unavailable for selection, its label should be grayed out to indicate its unavailability.

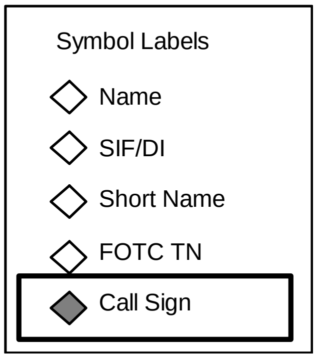


Figure 6-4. Example radio buttons (from the JMCIS style guide).

In some cases, it may be appropriate for users to select none of the options available in a group of radio buttons. In this case, a radio button labeled None should be provided as an option. A group of radio buttons should not be designed so that users are able to deselect all of the buttons in the group.

A group of radio buttons should provide no more than eight alternatives and should include a title that describes the type of information being presented. The preferred orientation for a group of radio buttons is vertical and left-aligned. However, if arranged horizontally, the radio buttons in the group should be separated sufficiently so that users will associate each diamond-shaped indicator with the correct description (i.e., so that the indicator is paired with the label to its right, not left).

6.2.2 Behavior

The Select button on the pointing device shall be used to select a radio button; when the location cursor is on a radio button, <Space> (and <Select> if available) shall select a radio button from the keyboard. If the radio button is in a window with a default action, <Enter> or <Return> shall execute the default action. The select state shall be indicated by highlighting (i.e., change in color) and by a change in appearance from raised to recessed.

When a radio button is selected, any previously selected radio button in the group of buttons shall be deselected. Users shall deselect a radio button by selecting a different radio button in the same group of buttons; clicking on a selected radio button shall not change its state to deselected. When users select a radio button, only the select state of the option should change; selecting a radio button should not initiate an action or display a dialog window.

6.3 CHECK BUTTONS

6.3.1 Appearance

A check button shall be used to set options in an application. A check button consists of a square-shaped indicator, followed by a label describing the option represented by the button. A check button is a nonexclusive setting, with a function that is similar to a toggle switch; selecting a check button shall toggle to the setting or state indicated by the label but should not normally invoke any further action. A check button (rather than two radio buttons) should be used if an option can only be set to on or off.

Check buttons can be used singly or in related groups. Figure 6-5 shows an example of a group of check buttons. Check buttons should be the same size whenever they appear in a window. The text label describing the option represented by a check button should be displayed in mixed case, with the first letter of the option capitalized. If a check button is unavailable for selection, its label should be grayed out to indicate its unavailability.

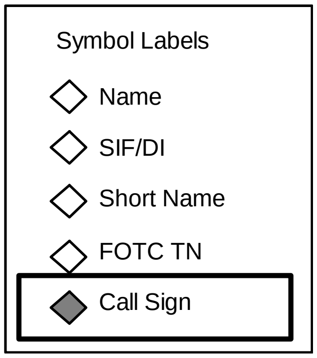


Figure 6-5. Example group of check buttons (from the JMCIS style guide).

The number of check buttons in a group should be limited to eight or less. As with radio buttons, the preferred orientation for check buttons is vertical. If a group of check buttons is arranged horizontally, they should be separated sufficiently so that users will associate each square-shaped indicator with the correct description (i.e., so that the indicator is paired with the label to its right, not left).

6.3.2 Behavior

The Select button on the pointing device shall be used to select a check button; when the location cursor is on a check button, <Space> (and <Select> if available) shall select it from the keyboard. If the check button is in a window with a default action, <Enter> or <Return> shall execute the default action. The select state of a check button is indicated by highlighting (i.e., a change in color) and a change in appearance from raised to recessed.

When a check button is selected, any previously selected check buttons shall remain selected. When users select a check button, only the select state of the option should change; selecting a check button should not initiate an action or display a dialog window. In addition, selecting one check button in a group should not affect the state of any other check button in the group. Users shall be able to select any or all of the check buttons in a group. Users shall deselect a check button by selecting it again.

6.4 TEXT FIELDS

6.4.1 Appearance

A text field, shown in figure 6-6, shall be used to enter and edit text. A text field may include a title which appears either to the left or above the field and describes what is to be entered. The label or title of a text field should be followed by a colon. A text field may also include scroll bars if the text being entered is longer than the field or extends beyond a single line.

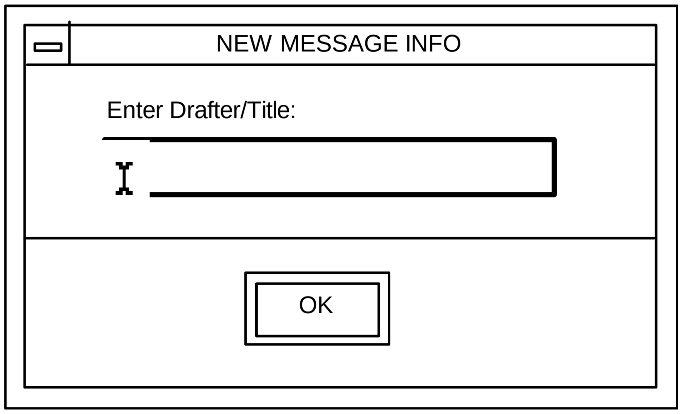


Figure 6-6. Example text field (from the JMCIS style guide).

6.4.1.1 Format of Text Fields

The title of a text field should include cues regarding format (e.g., Date [YYMMDD]) as appropriate. When a unit of measurement (e.g., feet, miles) is always associated with a field, it should be displayed as part of the title and not have to be entered by users. The format of a text field should be consistent with users' expectations (e.g., users should type numbers as numerals and not as words). Whenever possible, the information called for in a field should be in "chunks" that are meaningful to users (e.g., whole words). Figure 6-7 provides an example of how a text field can provide cues regarding format.

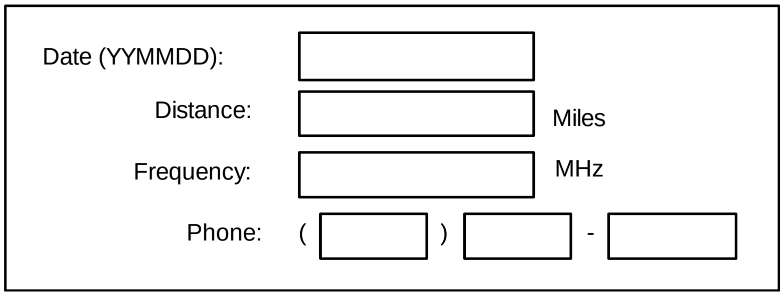


Figure 6-7. Example field labels indicating format (from the Operations Directorate GUI style guide).

The title should be designed so that users can easily distinguish it from the text field. The words used in the title of a text field should be clearly different from those used in the titles of other text fields; titles that contain the same word(s) are more likely to be confused by users. When the titles of a set of text fields are highly redundant (e.g., ship name, ship UIC, ship homeport), developers should consider renaming the text fields to remove the common word from the title (e.g., name, UIC, homeport) and placing it instead in the heading (e.g., ship information) that describes the set of fields.

A text field should indicate the basic features of the entry required. In addition, the field should be long enough for users to enter the information required. If the information being entered is a fixed length, then the field should be the same length as the information. If the information being entered varies in length, the field should be at least as long as the longest information. If window space is limited, a text field may contain scroll bars. Finally, the text field should include a decimal point if users are required to enter numeric data in decimal format.

If users have to enter a string of characters (that is not a word) longer than five to seven characters, the field should be broken into subgroups of three to four characters and separated by a space or delimiter, as shown in figure 6-8. Whenever possible, the subgroups should be meaningful to users (e.g., day, month, and year in a date; the three parts of a Social Security number). In addition, routine or default data, data already known by the application, or data that can be computed by the application should be automatically entered in a field. For example, if fields are provided for the start date, end date, and duration of a mission, the user should have to enter only two of the values and the application should calculate the third.

Delimiters not required:	Delimiters required:	
Do this:	Do this:	Not this:
Day: ☐	25/5/86	060589
Month: ☐	25-12-90	91|05|07
Year: ☐	15:3:92	7.5.90

Figure 6-8. Examples of text field delimiters (from MIL-HDBK-761A).

Text fields should be identified as mandatory or optional; for example, a visual cue such as an asterisk could precede the label for mandatory fields, with an explanation of the meaning assigned to the symbol included in the message area of the window, if one is provided, or in the help information available on the window.[38]

6.4.1.2 Distinguishing Text Entry Areas from Noneditable Text

If text can be edited, it shall be displayed in a text field. Noneditable text includes static information (i.e., labels such as titles, headings, and directions) as well as system-generated dynamic data that cannot be modified by the user. Static text shall be presented directly in the window, while system-generated data that is not editable can be presented either directly in the window or in a noneditable text field. Applications should ensure that each type of text has a distinctive appearance so that users can easily differentiate among them. The specifications presented here use color (see section 8.3.1) to provide this distinction. If a noneditable text field is used in a window, the field should have a different appearance (e.g., different background color) so that it can be easily distinguished from an editable field. In addition, when users move the pointer into a noneditable text field, the pointer should not change to an I-beam shape. When users click the Select button on the pointing device in a noneditable text field, the area should not change appearance (i.e., users should not be able to assign keyboard focus to noneditable text), and the text cursor should not appear in the field.

6.4.2 Behavior

6.4.2.1 Text Entry in a Text Field

Users should perform text entry as described in section 2.3.3. In addition, editing commands such as Cut/Copy and Paste (see section 3.4.2) should be available if these functions are appropriate for the task being performed. The default for text entry in Motif is insert mode, with <Insert> available to toggle between insert and replace mode in a text field.

When users type variable-length information in a text field, the information should be automatically justified or truncated; users should not have to enter leading characters to fill the space available (see figure 6-9). In addition, users should be able to enter numeric data from either the keyboard or the numeric keypad, especially when users have substantial amounts of numeric data to enter. Finally, the amount of data users have to enter in a text field should be minimized; for example, users should be able to enter an abbreviation rather than an entire word and should not have to enter the unit of measurement associated with a number value. Whenever possible, applications should perform automatic entry of data into a text field (e.g., prefill a date/time group field with the current date and time, or a lat/long field with the current position of the user's ship) and support position hooking (i.e., allow users to click on a geographic location and have the coordinates of the position entered into a lat/long field). Windows that support

[38]. Different text fonts should not be used to distinguish between mandatory and optional fields if the system provides users with the flexibility to select the text font and size in which to display window contents.

position hooking should indicate that this feature is available (e.g., in the label for the text field, with a special symbol next to the field).

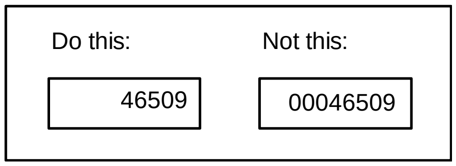


Figure 6-9. Example of automatic justification during text entry (from the DoD style guide).

When users move keyboard focus to a mandatory field, the application should not require that they enter data in the field before moving to the next field. Similarly, an application should not require users to correct an error in data entry before moving to another field, especially if the "error" may be fixed as a result of subsequent data entries. Users should not be constrained to performing text entry or doing error correction in a lock-step manner prescribed by the application. Instead, they should be allowed to enter data in whatever order they choose, with errors flagged when they occur. Users should be unable to commit the data (i.e., to execute a Save action) until they have completed all mandatory data entry and corrected all input errors.

When users make an error, they should receive feedback (e.g., a change in the color of the text field background, an audio signal) that an error has occurred. Users should be able to fix errors by editing individual characters in the field, rather than having to erase and retype the entire field. If users are entering data in multiple text fields and the fields are unrelated to one other (i.e., if correct input in one field does not depend on the input in another), then users should control when error checking will occur. Alternatively, when fields are interdependent, error checking should occur on a field-by-field basis so that users can detect and correct errors on a timely basis.

In cases where the number of valid entries in a text field is limited, developers may choose to provide a pop-up menu within a field that lists the entries from which users can select to fill the field. Alternatively, developers should consider creating one of the combination controls defined in section 6.8.

6.4.2.2 Navigation Between and Within Text Fields

Users should move the text cursor between text fields by pressing <Tab> or the arrow keys;[39] applications should not automatically tab to the next field. When users press one

[39]. The DoD style guide recommends that <Return>, <Enter>, <Tab>, and the arrow keys be used to move the text cursor between and within text fields from the keyboard. The specifications presented here, which follow the Motif Style Guide, indicate that when the location cursor is on a single-line text field, <Return> and <Enter> execute the default action (if one is available) in a window.

of these keys to move the text cursor to another text field, the cursor should appear at the left end of the field. Users should also be able to place the text cursor in any of the text in a text field by positioning the pointer on the text and pressing the Select button on the pointing device. The size of the text cursor should change as needed to match the size of the text font in which it is placed.

In a multi-line text field, the arrow keys on the keyboard shall move the text cursor one increment (i.e., one line or one character) in the specified direction, and <Ctrl> in combination with the arrow keys shall move the slider one large increment (i.e., one word or one paragraph) in the specified direction. In addition, <Home> and <End> shall move the text cursor to the beginning and end of a line, <Ctrl><Home> and <Ctrl><End> shall move the text cursor to the beginning and end of the text. If multi-line text is in a window with a default action, <Enter> or <Ctrl><Return> shall execute the default action.

Autotabbing may be used only when data such as date, time, latitude, and longitude are broken into smaller groups of characters, with each group entered in a separate text field (e.g. in the Phone field in figure 6-7). In this case, autotabbing may be used since users consider the characters to be a single data value and expect to enter the data without the need to tab between the fields; while separate text fields are intended to improve readability and minimize the opportunity for error, they should not interfere with efficient data entry by users.

6.5 LIST BOXES

6.5.1 Appearance

A list box such as the one shown in figure 6-10 shall be used to present a set of items from which to choose.[40] The items in the list box shall be displayed vertically, with one item per line. If the list box includes a title, it should describe the purpose or contents of the list; the title should appear above the box and not be followed by a colon. A vertical scroll bar shall appear to the right of the items in the list box; users shall be able to move the slider in the scroll bar if the items in the list exceed the space available in the box. The list box should scroll only in response to a user action (e.g., using the scroll bar, conducting a speed or incremental search) and should not scroll automatically (e.g., whenever the content of a list is updated through an automatic process).

[40]. Motif defines a list box as containing a set of user-selectable items. Developers seeking to define a list box that contains view-only items should follow the guidelines presented here concerning the appearance and behavior of noneditable text.

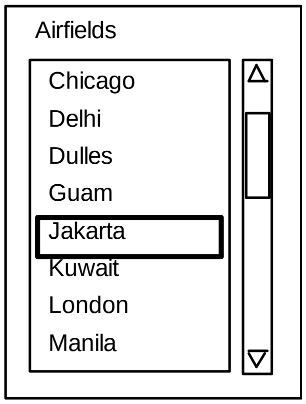


Figure 6-10. Example list box (from the JMCIS style guide).

The size of a list box depends on the amount of space available in the window in which it will be displayed. In general, a list box should be large enough to display six to eight items at a time, or all of the items if there are fewer than six. A list box should be wide enough so that users can read all of the items without scrolling horizontally; if items differ significantly in length, then the list box should be wide enough to display the items of average length and include a horizontal scroll bar to allow users to read the longer items.

Depending on the functionality provided by a window, developers may choose to designate one of the items in the list as the default so that when the list is first displayed, the default item is selected (and is highlighted). The default may be the first item in the list, the item users are most likely to select, or the item(s) that users had selected when the window was previously displayed (e.g., if the selection was saved when the window was closed).

The items in a list box should be presented in sequential order based on the nature of the items and the sequence in which users expect the items to occur (e.g., chronological, alphabetical, sequential, functional, by importance). For example, a list of port names should be ordered alphabetically, and a list of messages by precedence, date-time group, or a combination of the two (e.g., date-time group within precedence).

6.5.2 Behavior

6.5.2.1 Adding Items to a List Box

When users add an item to a list box, the item should appear in its correct position within the list (e.g., in numerical or alphabetical order) rather than at the end of the list. Users should be able to quickly determine where in a list to expect a new item to appear.

6.5.2.2 Searching Items in a List Box

Users shall be able to search items in a list box with either the pointing device or from the keyboard. For example, users shall move the slider on the scroll bar until the item appears in the box. The arrow keys shall also scroll the items in the list. In addition,

<Ctrl><Begin> and <Ctrl><End> shall scroll to the beginning and end of the list, and <PageUp>, <PageDown>, <PageLeft> (or <Ctrl><PageUp>), and <PageRight> (or <Ctrl><PageDown>) shall page through the list in the direction indicated.

Applications should provide a speed search capability in all list boxes.[41] If applications provide this capability, it shall function in the manner described here. Users shall place the location cursor on the list. When the list box has focus and users type a character, the list shall scroll to the first instance of an item that begins with that letter and the location cursor shall move to that item. When users type the character again, the list shall scroll (as needed) and the location cursor shall move to the next item that starts with the character. If the character typed does not match any of the items in the list, the location cursor shall not move, and users should receive feedback (e.g., an auditory signal, a message in the message bar at the bottom of the window, or a message window) to indicate that no match was found.

When the list box contains a large number of items (e.g., 50 or more), applications should provide a text field with the list box, as in figure 6-11, so that users can perform an incremental search. If applications provide this capability, it shall function in the manner described here. To perform this type of search, users shall type the first few letters of the item desired in the text field. If appropriate, users should be able to enter wild card characters to search for specific text patterns, as described in section 8.4.7. When they press <Return>, the list scrolls to the first occurrence of an item that matches the letters. Users then use the scroll bars to scan through this part of the list to locate the item desired. If the character(s) typed do not match any of the items in the list, users should receive feedback (e.g., as indicated in the previous paragraph) to indicate that no match was found. If possible, an incremental search capability should be case-insensitive. However, if the search must be case sensitive, then this information should be provided to users, as in figure 6-12.

[41]. This capability is not currently supported by Motif. However, implementation of a search capability is recommended, especially when users are presented with lists that contain a large number of items.

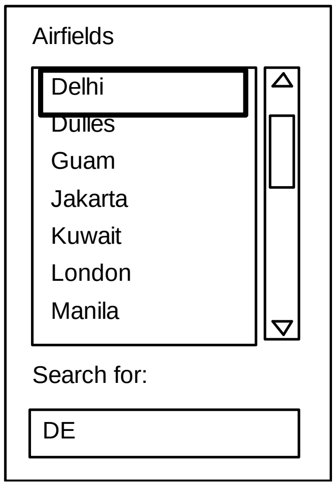


Figure 6-11. Example list and text field used in an incremental search (from the JMCIS style guide).

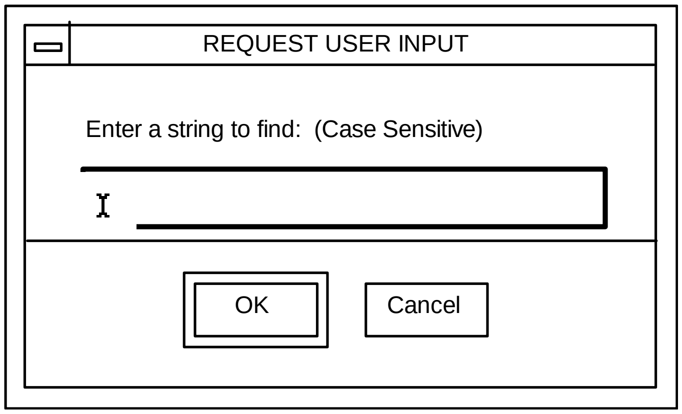


Figure 6-12. Example of a case-sensitive search (from the JMCIS style guide).

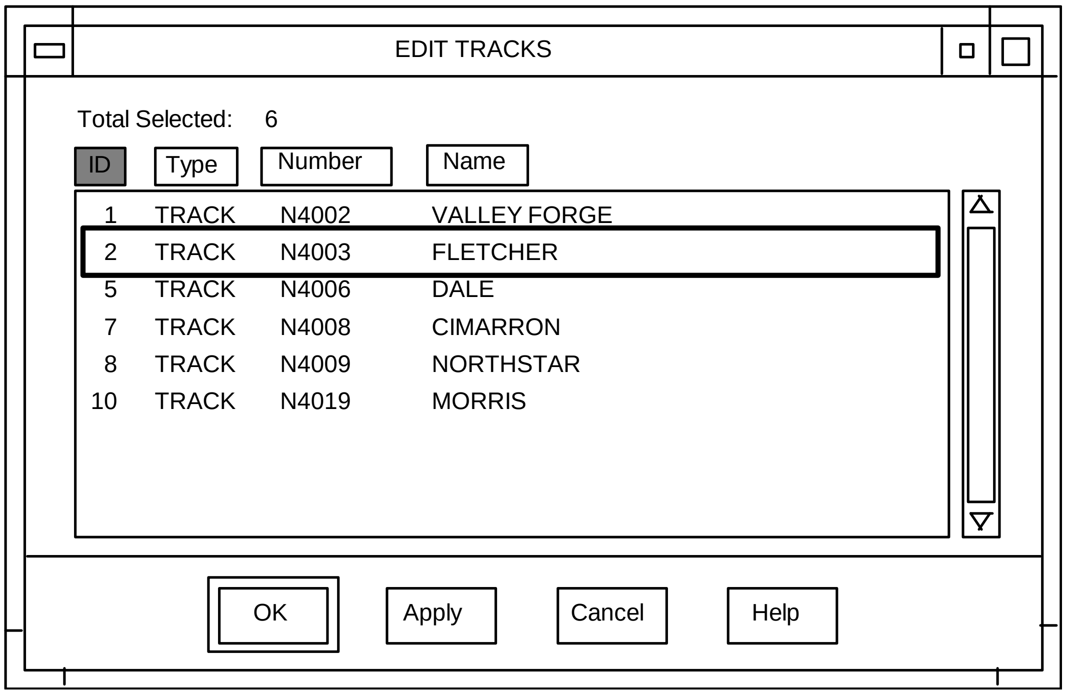
6.5.2.3 Selecting Items in a List Box

The Select button on the pointing device shall be used to select an item in a list box. When the location cursor is on an item, <Space> (and <Select> if available) shall select it from the keyboard. The select state of a list item shall be indicated by reverse video (i.e., reversing the foreground and background color of the item). If the list box is in a window with a default action, users shall double click the Select button on the pointing device to choose an item and execute the default action. Selecting an item in a list box should not affect the order of the items in the list; e.g., when an item is selected, it should not be moved to the top of the list.

If more than one item in the list can be selected, users shall follow the methods described in tables 3-1 and 3-2 to select multiple items. In addition, developers should consider providing this information in a label that is adjacent to the list box, and feedback (e.g., a counter) should be provided to the user (e.g., in the information area of the window) on the number of items that have been selected.

6.5.3 Multi-Column List Boxes

A series of list boxes may be arranged horizontally in a row to form a multi-column list box or matrix, as shown in figure 6-13. The list box titles serve as the column headings in the matrix, and the items in each list form the records that appear in the rows. Applications should provide a consistent approach for sorting records that appear in this type of matrix. For example, the headings that can be sorted can appear as buttons (see section 8.1.3.4) so that clicking on the heading sorts the records in an order based on the items in that column. If additional sort variations are needed, they should be made available within the window (e.g., as an option in a pull-down menu or as a push button). If desired, applications can provide a speed search capability within this type of list box so that users can both sort the items in the list and then execute a speed search to scroll to the first instance of an item that begins with a particular letter.



Total Selected: 6

ID	Type	Number	Name
1	TRACK	N4002	VALLEY FORGE
2	TRACK	N4003	FLETCHER
5	TRACK	N4006	DALE
7	TRACK	N4008	CIMARRON
8	TRACK	N4009	NORTHSTAR
10	TRACK	N4019	MORRIS

Figure 6-13. Example list matrix (from the JMCIS style guide).

6.5.4 List-to-List Transfer

A series of list boxes may be arranged to form a list-to-list transfer window, as shown in figure 6-14. This window contains a source list on the left and a destination list on the right, separated by two push buttons that allow items to be transferred between the two lists. The push buttons may contain text labels (e.g., Add, Remove) or left and right arrows indicating the direction of the move. When keyboard focus is on the source list (and one or more of the items is selected), Add should be displayed as the default and Remove should be disabled. Conversely, when focus is on the destination list, Add should be disabled and Remove shown as the default. Users should be able to transfer more than one item at a time between lists but not to transfer multiple instances of the same item to the destination list. The window may include radio and check buttons that allow users to modify the contents of the source list (e.g., to limit the items in the source list to those with specific features).

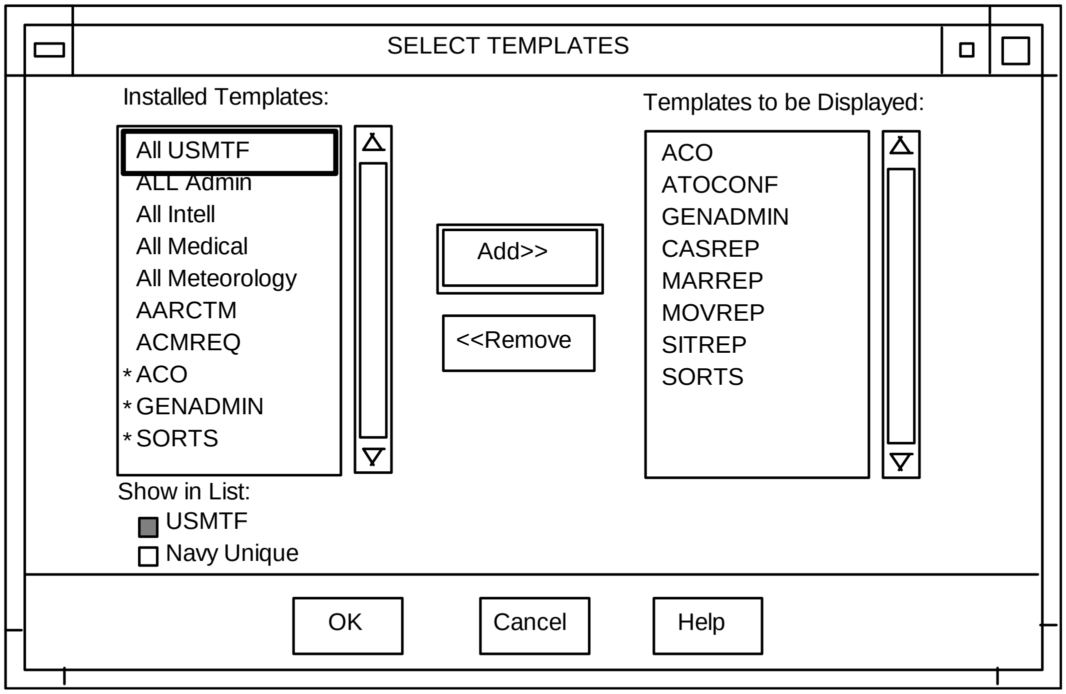


Figure 6-14. Example list-to-list transfer window (from the TBM style guide).

Depending on the nature of the transfer task, an item in the source list may be either copied or moved when users transfer it to the destination list. In the former case, the item should be marked (e.g., with an asterisk) to indicate that it has been transferred. The mark should be removed when users transfer the item from the destination list back to the source list.

6.6 SCROLL BARS

6.6.1 Appearance

Scroll bars shall be used to view textual or graphic information when it exceeds the space available to display it. Vertical scroll bars shall control backward and forward movement through the information; horizontal scroll bars shall control left and right movement.

A scroll bar shall contain the components shown in Figure 6-15.[42] The trough region is the background of the scroll bar and shall represent visually the length of the information that users can scroll. The slider shall represent the window through which users view the information. The stepper arrows enable users to scroll incrementally through the information, and shall indicate the direction of the scrolling movement. The slider shall move back and forth in the trough region and show the position of the currently displayed information relative to the entire amount. Users shall be able to scroll to the top or the bottom of the information but not beyond.

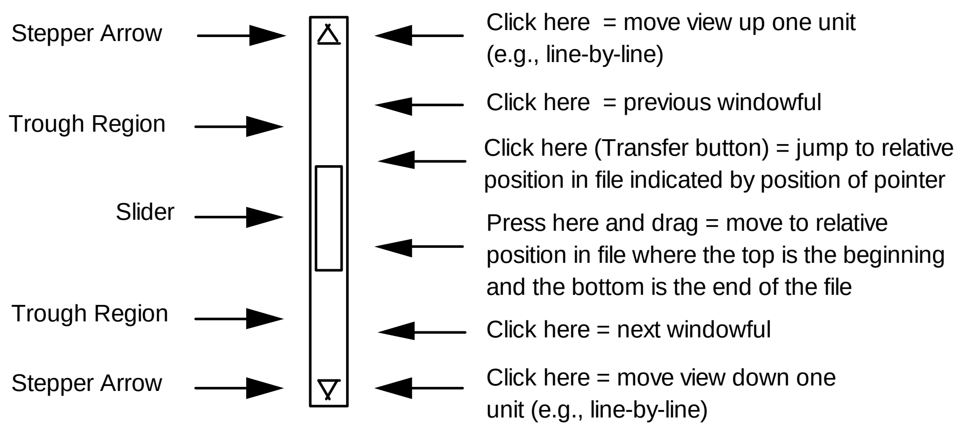

Stepper Arrow
Trough Region
Slider
Trough Region
Stepper Arrow
Click here = move view up one unit (e.g., line-by-line)
Click here = previous windowful
Click here (Transfer button) = jump to relative position in file indicated by position of pointer
Press here and drag = move to relative position in file where the top is the beginning and the bottom is the end of the file
Click here = next windowful
Click here = move view down one unit (e.g., line-by-line)

Figure 6-15. Components of a scroll bar (from the IDHS style guide).

The relative position of the slider represents the relative position of the information currently displayed in the window. For example, the middle list box in figure 6-16 shows all of the items in the list so the slider fills the trough region. In the other two list boxes, the position of the slider indicates that the items displayed are from the beginning of the list, and length of the slider indicates that the list is longer than the space available in the box.

[42]. Variations in the standard scroll bar (e.g., dual-headed stepper arrows at both ends of the bar) will be addressed in a future version of this document.

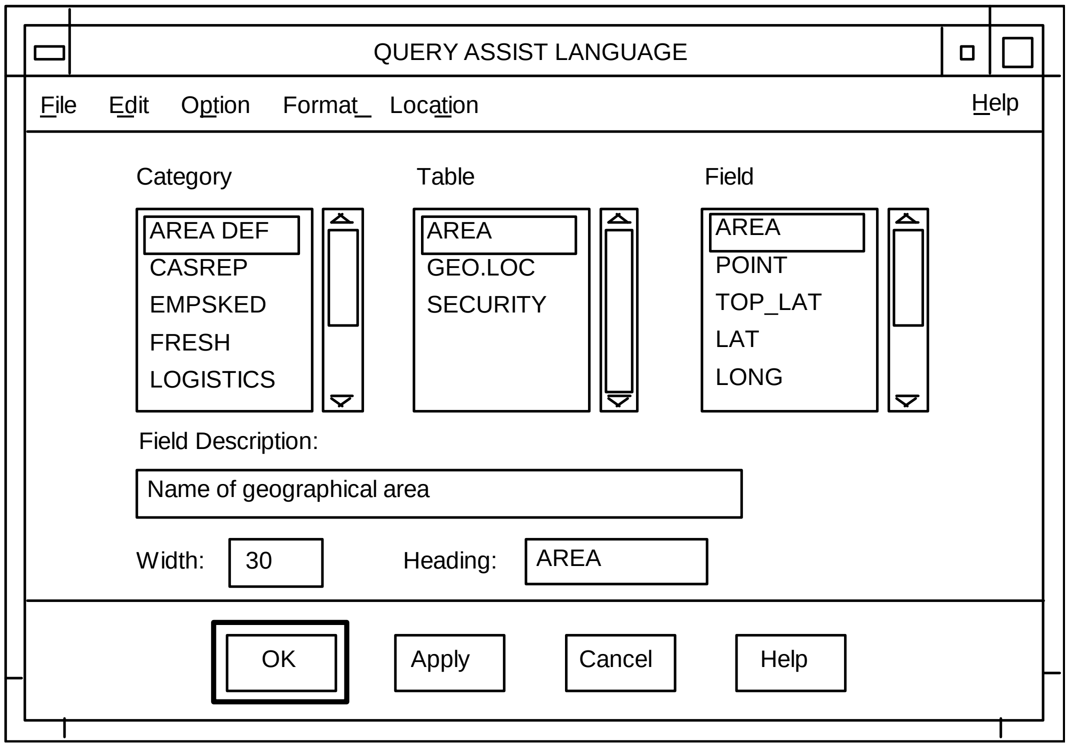


Figure 6-16. Examples of slider position within trough regions (from the JMCIS style guide).

6.6.2 Behavior

Users shall press on a stepper arrow with the Select button to move the slider in one-unit increments (e.g., one line or column) in the direction indicated by the arrow. Users shall press on the trough region with the Select button to move the slider one window length (or width) minus one unit (for overlap) at a time. When users drag the slider with the Select button, the slider shall track the position of the pointer. If users press on the trough with the Transfer button, the slider moves to the position where the button press occurred; users can then drag the slider with the Transfer button to the position desired. <Cancel> shall return the slider to its position before the sliding operation began.

When users position the pointer on a scrollable control and drag the pointer (i.e., with the Select button pressed) outside the control, the information should scroll in the direction that the pointer is being dragged (i.e., autoscrolling). The information should scroll at the same speed as when users press on a stepper arrow. Scrolling should stop when users move the pointer back into the control or when users release the Select button on the pointing device.

When the location cursor is on a scroll bar, the arrow keys shall move the slider one increment in the specified direction. <Ctrl> in combination with the arrow keys shall move the slider one large increment, and <PageUp>, <PageDown>, <PageLeft> (or <Ctrl><PageUp>), and <PageRight> (or <Ctrl><PageDown>) shall page in the specified direction. In addition, <Ctrl><Begin> and <Ctrl><End> shall scroll to the beginning and end of the scrollable region.

6.7 SCALES

6.7.1 Appearance

A scale (as shown in figure 6-17) shall be used to set or display a value in a range. Users shall display a value by adjusting a slider (bar or arrow) to a specified position along a line. A scale shall consist of a trough region, a slider marking the currently chosen scale value, and a label above or next to the slider showing the current value of the scale. In addition, the scale bar should include tick marks representing the range of available values and be labeled with the minimum and maximum values for the scale.

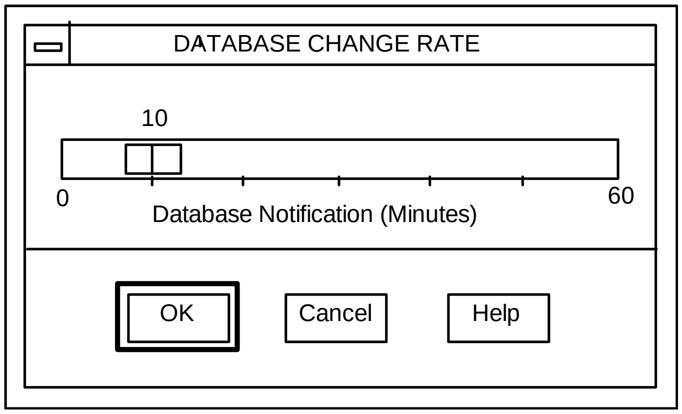


Figure 6-17. Example scale (from the JMCIS style guide).

6.7.2 Behavior

The movement of the slider in a scale is similar to the movement of a scroll bar, except that the scale also indicates the value. If the scale has arrow buttons, users shall press the Select button on an arrow to move the slider one unit at a time in the direction indicated by the arrow. Users shall press on the scale bar with the Selectbutton to move one large increment (defined by the tick marks, if provided) at a time in the direction indicated. Users shall drag the slider with the Select button to position the slider to reflect the scale value desired. If users press on the trough with the Transfer button, the slider moves to the position where the button press occurred; users can then drag the slider with the Transfer button to reflect the scale value desired. <Cancel> shall return the slider to its position before users began the sliding operation.

The arrow keys on the keyboard shall move the slider one increment in the specified direction. <Ctrl> in combination with the arrow keys shall move the slider one large increment, and <Ctrl><Begin> and <Ctrl><End> shall move the slider to the minimum and maximum scale values.

6.7.3 Gauges

A gauge is a type of scale used to present values that users cannot change. For example, a working message window may include a gauge to provide dynamic feedback to users on the percent of a process that is complete. If a gauge is used, it should contain a scale indicating the units of measurement represented by the gauge, and the trough region should be filled dynamically to indicate the relative amount of processing completed. If the exact percentage is important, a gauge can include a label indicating the current percentage value, and both the trough and label can be updated dynamically. Because the values displayed by a gauge cannot be changed, a gauge should not include a slider or arrow buttons.

6.8 COMBINATION CONTROLS

Although combination controls are not available in version 1.2 of Motif, developers choosing to incorporate public domain versions of these controls or develop their own should do so in accordance with the general guidelines presented here. More detailed specifications concerning the appearance and behavior of combination controls will be provided when the software configuration defined in section 1.3.1 is upgraded to a version of Motif that includes these controls.

6.8.1 Combination Boxes

A combination box consists of an editable single-line text field and a list box displayed immediately below the text field, as shown in figure 6-18. Users can either select one of the items from the list to display in the text field, or type directly in the field. When users select an item from the list, it should replace any text in the field. When users type in the text field, the text entered does not have to match one of the items in the list. In addition, text entry should not affect the contents of the list; when users type in the text field, the text is not added as a new item to the list.

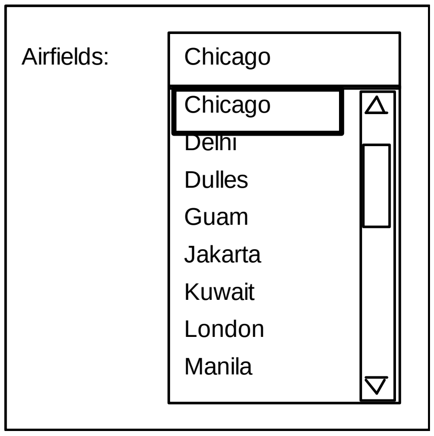


Figure 6-18. Example combination box (from the JMCIS style guide).

The list in a combination box can contain any number of items. The list should be large enough to display six to eight items at a time, or all of the items if there are fewer than six. A vertical scroll bar should be provided when the list is too long for all of the items to be visible in the window. The combination box should be wide enough that users can read all of the items in the list, with the text field the same width as the list. Items in the list should be ordered according to the guidelines presented in section 6.5.1. When a combination box is initially displayed, the text field can either be empty or pre-filled with a default item from the list. In the latter case, when the combination box receives keyboard focus, the default entry should be highlighted so that text typed by users overwrites this text.

6.8.2 Drop-Down Combination Boxes

A drop-down combination box (see figure 6-19) consists of an editable single-line text field, a down-pointing arrow button, and a list box that is displayed when the arrow button is depressed. Users can either view the list and select one of the items to display in the text field, or type directly in the field. Guidelines on the appearance and behavior of combination boxes also apply to drop-down combination boxes.

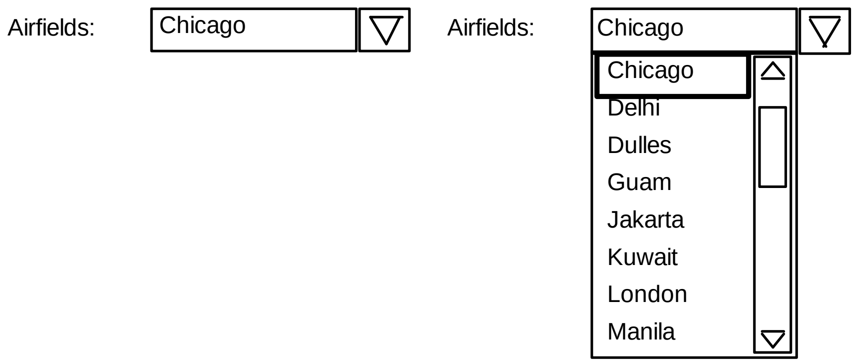


Figure 6-19. Example drop-down combination box (from the JMCIS style guide).

6.8.3 Spin Buttons

A spin button is a specialized text field for entering a limited number (less than 20) of discrete, ordered values (e.g., months of the year). A spin button consists of a single-line text field, with up- and down-pointing arrow buttons to the right of the field, as shown in figure 6-20. The text field can be editable (e.g., if the list of "spin" entries does not include all possible values) or noneditable (e.g., if the list is short and includes all possible values). When a spin button is displayed, the text field should contain a default value. Users click on the up and down arrows to "spin" (i.e., increase or decrease) the entry in the field. When the largest or smallest value is reached, the entries should wrap so that users can cycle continuous through the range of values. If the text field is editable, users can also type a value directly in the field.

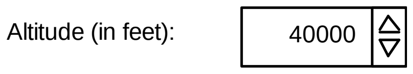


Figure 6-20. Example spin button (from the TBM style guide).

If spin buttons are used to enter data such as date/time group or lat/long, separate spin buttons should be used for each part of the entry. Spin buttons may be combined with standard text fields for data entry (e.g., separate spin buttons for month and day, with a text field for typing the year).

6.9 STANDARD AND NONSTANDARD CONTROLS

6.9.1 Consistent Appearance and Behavior

All of the controls in an application window should be identifiable solely on the basis of their appearance, and all controls with the same function should have the same appearance. Users should be able to distinguish between controls that are similar in shape (e.g., a push button and an option menu) on the basis of distinctive visual cues (e.g., an option menu should include a bar graphic). Users should not have to select a control in order to determine what it is and how it behaves.

When text or graphics is presented in an application window, it should be clearly different in appearance from standard controls. For example, one method of drawing attention to information in a window (e.g., the text in a heading) is to place a frame around it. However, a frame border should not be used if it results in an object with the same appearance as a window control (e.g., a “shadow out” frame has the same shape as a push button).

6.9.2 Using Nonstandard Controls

Developers should use existing controls whenever possible to provide application functionality. However, developers can modify the features of a control if the modification is needed to support efficient task performance by users. If a nonstandard control is used, it should possess as much as possible of the standard appearance and behavior of the control from which it was derived.[43]

6.9.3 Adapting Controls When Using Commercial Software

As indicated in section 1.6.2, applications that make use of COTS software shall configure these products to be compliant with the specifications presented here insofar as possible. When doing so, developers should ensure that they do not modify existing controls in

[43]. Future versions of this document will identify the set of nonstandard controls allowed within these specifications.[44]. Compliance with the style guidelines for the Compartmented Mode Workstation, which is included as an appendix to the DoD style guide, will be addressed in a future version of this document.

ways that conflict with the appearance and behavior defined in these specifications. For example, if a COTS product does not include a control such as a push button, developers should not define an object with the appearance of a text field and the behavior of a push button in order to provide this functionality. Likewise, developers should not modify the pull-down menus in a window so that the menu titles function as push buttons; this type of modification is inconsistent with both the normal behavior of menu titles (i.e., when selected, a pull-down menu is displayed) and with Motif style guidelines on the placement of objects in windows (i.e., push buttons are not normally placed in a menu bar). When selecting a COTS product, developers should consider the extent to which the product can be adapted to fit the style defined by these specifications. If significant discrepancies (such as those described above) will result, the product may be inappropriate for use given the inconsistencies in “look and feel” that will result if it is integrated within a system.

7.0 SYSTEM-LEVEL WINDOWS

7.1 SYSTEM LOGIN

7.1.1 Login Procedure

Each system shall implement a login procedure that users must complete before they can access any system functions. A system may make available at login only those applications to which the user is allowed access, or may require users to log in to individual applications or groups of applications. If the system is unavailable for login, a message should be displayed, whenever possible, indicating the system status and when the system will become available.

7.1.2 The Login Window

A login window shall appear on the screen when users begin a session on a system. This window, shown in figure 7-1, shall contain two text entry fields for users to enter their user identification and password. The text fields shall be longer than the information that user will type in them so that their size does not give unauthorized users a clue as to the number of characters required for system access. The appearance and behavior of the objects in the login window (i.e., background, text fields, text) shall be consistent with the specifications in tables 7-1 and 8-1.

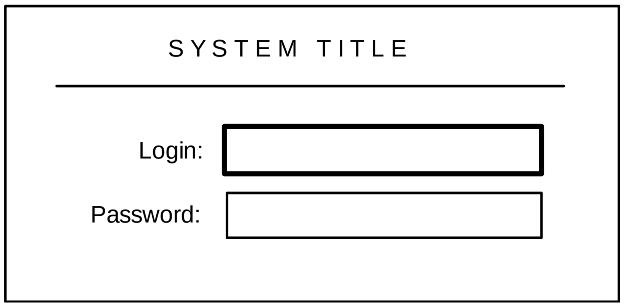


Figure 7-1. Example login window (from the JMCIS style guide).

Users must enter a valid identification and password before a session is initiated. If users enter an invalid identification or password, an error message shall appear in the window. Users who fail repeatedly to log on successfully shall be locked out of the system and informed that they should contact the system administrator.

7.1.3 System Start-Up

System start-up occurs after users complete the login process successfully. During start-up, the system should display a message window indicating its unavailability, change the pointer shape to a watch, and disable input from the keyboard and pointing device. The message should disappear, the pointer should return to its standard shape, and the input devices should be enabled when start-up is complete and the system is available for input from users. If appropriate, the system should provide messages indicating system status, e.g., average system response time (if system response is affected by the number of users) and known periods of unavailability.

Each system shall specify the initial appearance of the system window following start-up. For example, the screen may be empty, with users selecting the initial application with which to interact. Alternatively, the system may define a default application that is opened during system start-up, with a primary window for the application (e.g., a map window) appearing automatically on the screen. In this case, the selection of a default application should be determined by the system and may vary depending on the type of user (e.g., the system administrator may have a different default application than a normal user).

7.1.4 Classification Markings

Each system shall provide appropriate classification markings following start-up that are controlled by the system and reflect the highest level of data currently displayed on the screen.[44] Each system shall implement the set of classification colors indicated in section 7.2.1. These colors shall be used as the background in the classification portion of the classification bar. The text shall be all upper-case letters, and the text font and size shall be consistent with the specifications for text in windows presented in section 8.4.1. All classification terms shall be spelled out, with caveats abbreviated in accordance with relevant security manuals and directives. In addition, there shall be no embedded spaces within words in the classification label (e.g., SECRET, not S E C R E T). Each system should provide an option for implementing alternate sets of classification colors as might be defined by the developing agency or the security officer at individual sites.

7.2 SYSTEM INTEGRATION USING A MENU-BASED MODEL

7.2.1 System Window Design

The system window[45] shall appear on the screen when system start-up is complete. This window shall cover the entire screen, as shown in figure 7-2, so that users cannot access operating system functions. The system window shall contain a classification bar and a system menu bar, both of which shall extend across the top of the window; a title bar containing the system name may be included but is not required. The current classification level shall appear in the middle of the classification bar; if desired, various status indicators (such as alerts) may be indicated at the left margin, and a digital date-time clock may be displayed at the right margin of the classification bar. The date and time should be displayed in the format described in section 8.4.6. The system menu bar shall list the titles of the menus available at the system level; these menus shall provide access to the set of application programs available within the system.

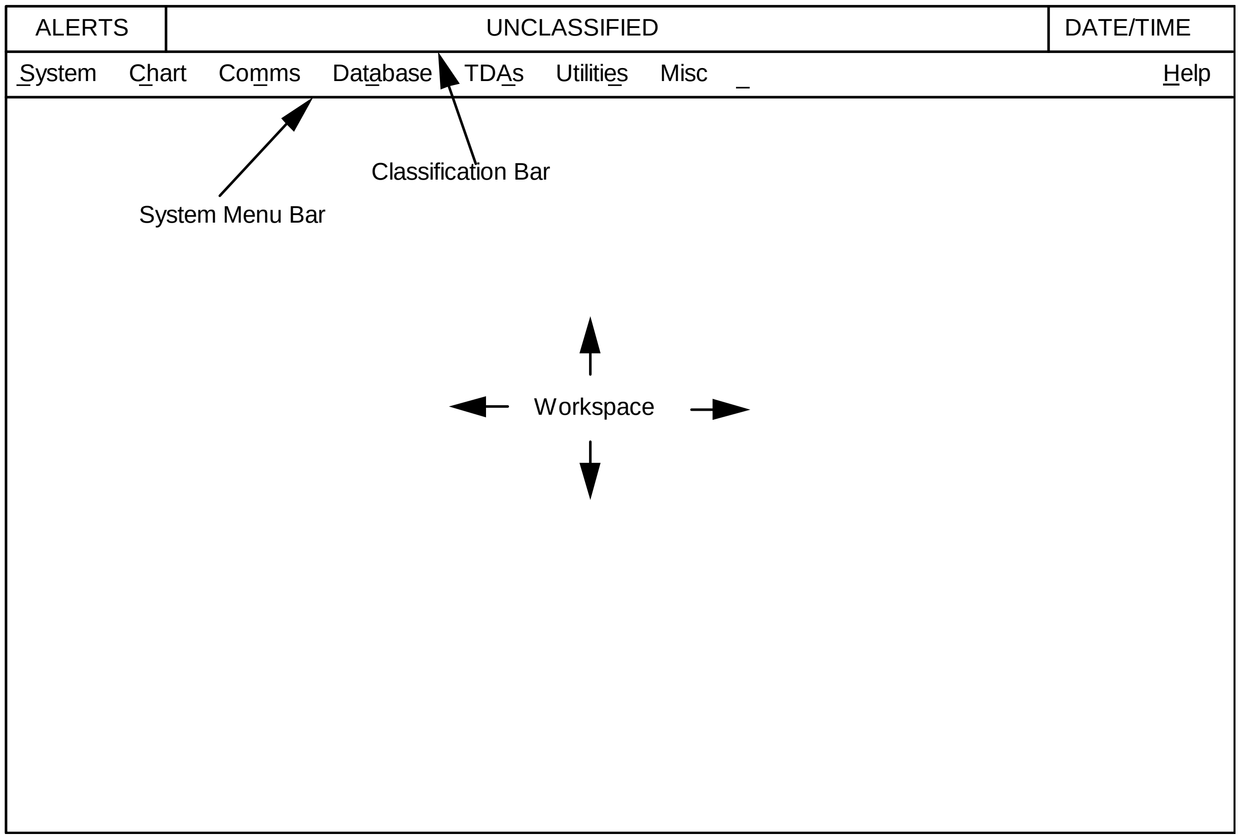


Figure 7-2. Example system window (from the JMCIS style guide).

[45]. The system window and the system-level functions to which it provides access correspond to the resource manager window and its functions as defined in the DoD style guide, except that the resource manager includes security markings based on Compartmented Mode Workstation guidelines included in the DoD style guide.

The remainder of the system window shall be available for displaying application windows. If desired, action icons may be used to provide access to frequently executed actions within a system (e.g., starting/stopping processes, changing input/output devices). Action icons that are common to all applications within a system should be placed along the left margin (from the top of the window downward) of the screen. When users minimize application windows, the icons should be displayed in the lower left corner of the screen. Each system should ensure that the window icons generated by each application contain graphic images that provide a visual indication of the functionality provided by the application.

The color set for the system window is presented in table 7-1 and figure 7-3. Appendix A contains the resource settings and RGB values for the color set.

Note to reviewers: Multiple color sets for the range of operational environments in which GCCS-based systems will be used need to be defined, with selection of a color set available as a user preference setting. Specific color names (and RGB values) for colors in the classification banner also need to be specified.

Table 7-1. Appearance of the system window.

Object	Appearance
Classification Bar	TBD background, TBD text (for alerts and date/time)
Unclassified banner	Green background, Black text
Confidential banner	Blue background, Black text
Secret banner	Red background, Black text
Top Secret banner	Orange background, Black text
System Menu Bar	TBD background, TBD text
Workspace	TBD background

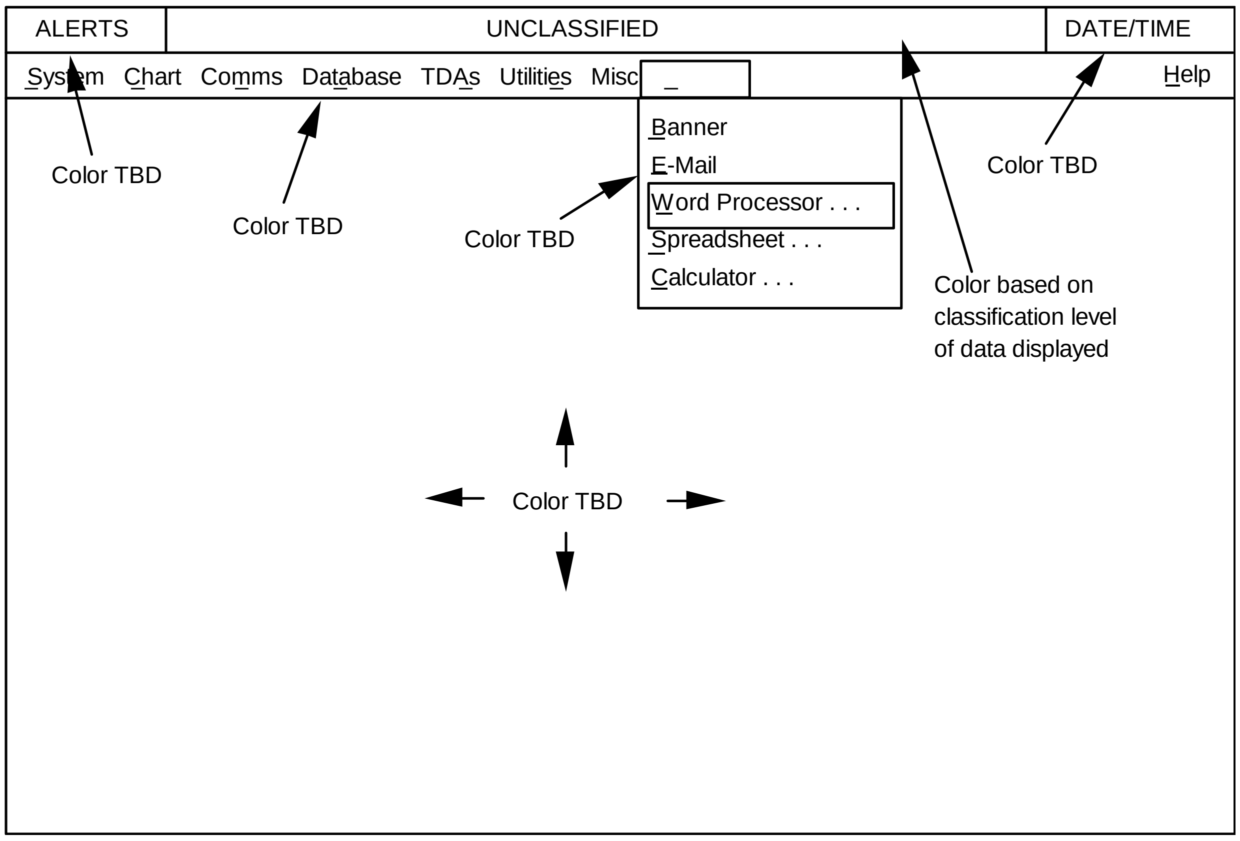


Figure 7-3. Example colors for system window.

The system window shall not be moved or resized, and the classification and system menu bars shall not be obscured by application windows that appear on the available screen space. For example, in figure 7-4, Window 1 covers the entire screen but does not hide the system menu bar. The system window shall be active at all times so that users working in an application can select help or any other system-level menu options appropriate to the application.

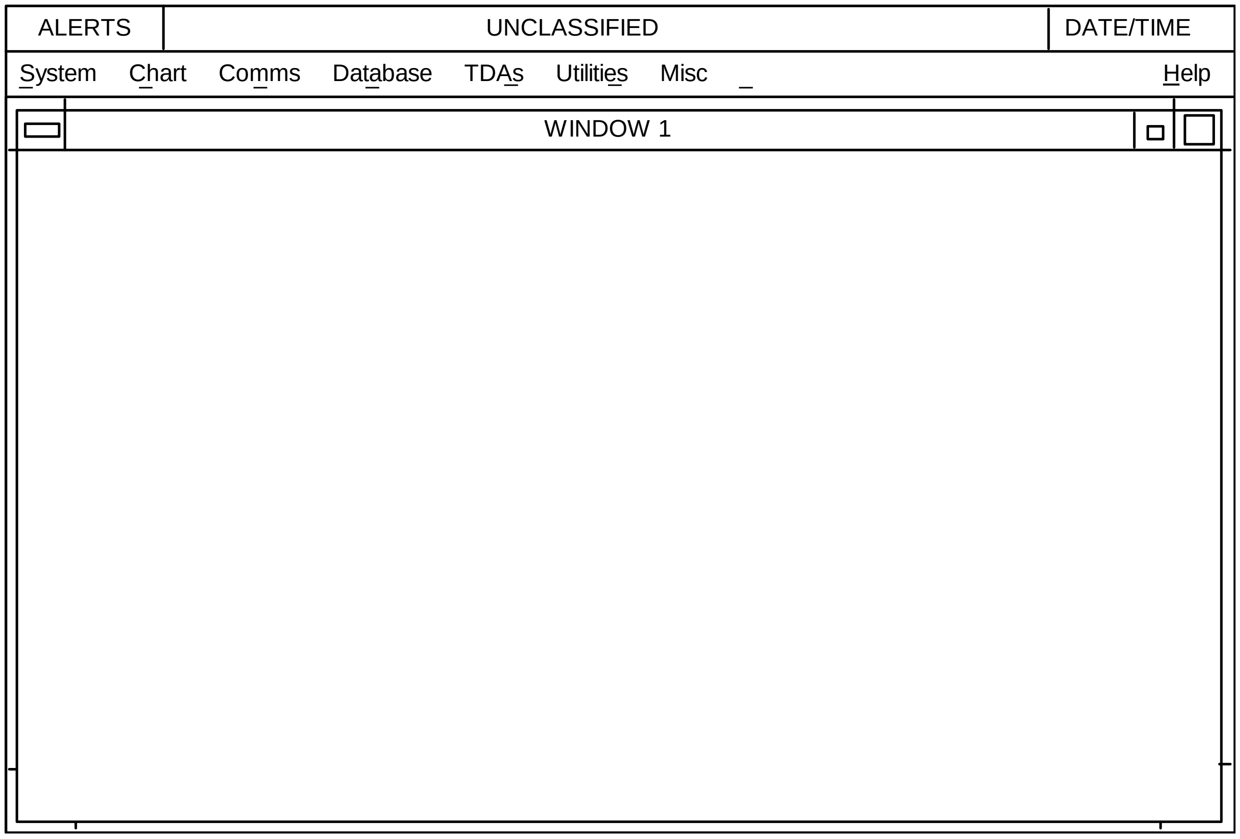


Figure 7-4. Maximum size of an application window (from the JMCIS style guide).

7.2.2 Integration of Applications into System-Level Menus

The integration of applications within a system should be performed according to the functionality provided by each application. This style guide distinguishes between applications that display geo-related tactical information from those that provide other tactical (e.g., communications, database) and generic functionality (e.g., COTS products). Applications of the latter type should be integrated into the appropriate menu in the system menu bar, while applications of the former type should be accessed from the menu bar of a single chart window (so that all of the information can be drawn on a common map). The chart window should provide its own core functions related to map manipulation, plot control, and track display, with other chart-related applications available from other menus in the window (e.g., a tactical decision aid that is executed only when contact data are displayed on a map).

Each system shall determine whether the functionality provided by an application will be available as an entire menu (e.g., a set of map display options), as a single menu option (e.g., a single decision aid or analysis tool), or as a group of options in a menu (e.g., specific message generation functions). In some cases, system integration may result in an application being decomposed and individual components distributed among multiple menus based on the functionality provided by the component, as shown in figure 7-5. This integration process results in a tighter coupling of applications than occurs in the desktop model, with users less likely to be aware of distinctions between applications as they work, for example, within a chart window.

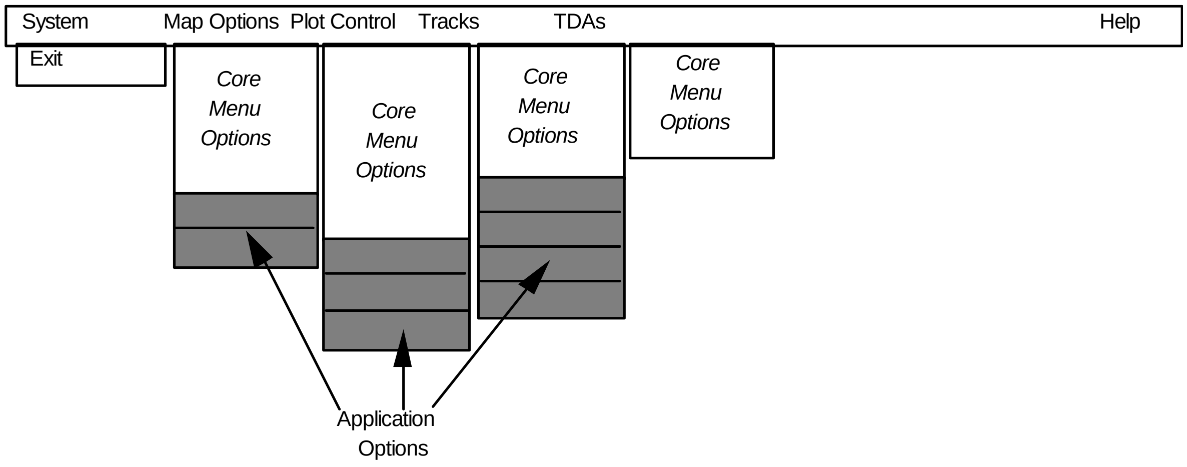


Figure 7-5. Example integration of application functions (from the JMCIS style guide).

Each system shall determine whether users should have access to an application or a function within an application (e.g., allow users to view but not modify or delete track data), and shall configure the contents of system-level menus accordingly (including deactivating keyboard accelerators associated with functions to which users are denied access). In this regard, it is recommended that systems disable any application-defined keyboard accelerators that would be available from a system-level menu since the system menu bar is always active and these accelerators may conflict with those executed by users while working within an application.

Each system shall determine the order in which application functions shall be available within system-level menus and ensure that this order is followed whenever the system is delivered to the user community.[46] In addition, it is possible for a system integrating a large number of software modules to create unusually long menus, sometimes extending beyond the

[46]. In Menu Executive, the order in which options are listed in menus is based on the order in which software modules are loaded. As a result, each system should define a standard sequence for loading modules and then adhere to this sequence whenever the system is built.[47]. If an application allows users to enter typed commands in a window, a command bar should be provided in the window; it should appear at the bottom of the window, above the message bar if one is included in the window.

bottom of the screen (especially when displayed in a large menu font). Each system should make use of cascading submenus, whenever possible, to reduce menu length. In addition, if the creation of lengthy menus cannot be prevented, each system shall ensure that each of these menus includes a capability (e.g., arrow buttons for scrolling the menu) for users to view and select the options that extend beyond the bottom of the screen. Finally, each system should implement navigation aids (such as those described in section 7.3.6) to provide users with the ability to tailor menu contents to the specific task being performed.

Each system should determine the extent to which the applications it contains perform common functions, and eventually integrate these functions so that they are accessible to users in a single window rather than available separately in each application. For example, applications that provide access to different sources of sensor data usually include a map window for displaying the information as part of the application. When these applications are initially integrated in a single system, users continue to interact with the information in separate map windows in each application. However, systems should eventually integrate application functions so that information from multiple sources is displayed in a common map window and users are able to perform the functions associated with each application in this window.

7.2.3 The System Menu Bar

7.2.3.1 Menu Bar Organization

The set of menu titles that appear in the system menu bar should describe the overall functionality provided by the system. The menu bar should contain no more than ten menu titles plus Help. The titles should begin at the left margin of the menu bar and extend rightward, with enough space between them to be easily read and to accommodate the longest options in each menu. Each menu title and menu option should include a mnemonic to support selection from the keyboard. If keyboard accelerators are defined in system-level menus, they should be selected carefully and in coordination with those defined by individual applications since the system menu bar is always active, allowing users to execute accelerators at any time, without regard to the application in which they are currently working.

In general, the leftmost menu titles should provide system, chart, and communications functions, the middle menu titles should provide functions that are specific to the particular system, followed by miscellaneous and support functions. Help should appear at the far right of the menu bar. Figure 7-2 is an example of a set of menu titles that might be included in a system menu bar.

7.2.3.2 Consistency in Menu Titles Between Systems

Each system should determine the extent to which it provides the same functionality as other systems. Whenever possible, the same system-level menu titles should be used when accessing this functionality. For example, the following menu titles are for two systems that provide both common and unique functionality:

System Chart Comms Database TDAs Misc Tools	Help
System Chart Comms Environmental Misc	Help

When the same application is shared by multiple systems, it should have the same name in each system and should be available in the same system-level menu. For example, if a Fuel Calc application is one of the TDAs available in two systems, it should be named Fuel Calc in both systems and appear in the TDAs menu in both systems.

7.2.3.3 Access to Menu Options Within an Application

Although a system defines the set of menu options that are included in each of its system-level menus, application developers should determine when these options are available for users to execute from within the application. When a system menu option is available within an application, the option should be displayed in normal text. Options that are inappropriate to execute from within the application should be dimmed. If all of the options in a menu are inappropriate, the menu title should also be dimmed.

Application developers should specify the number of instances of the application that can be running at one time. If only one instance can be running (e.g., on different screens), the menu option that launched the application should be dimmed while users are interacting with the application. If multiple instances can be running, the menu option should return to its normal appearance after it has been selected to indicate that users can select it again if desired.

When users are working in an application, they should be able to select Help from the system menu bar at any time. In addition, users should be able to browse all of the menu titles in the system menu bar even if all of the options in the menu are unavailable from within an application.

7.3 SYSTEM INTEGRATION USING A DESKTOP MODEL

Note to reviewers: This section will define the system window when a desktop model is used, address the integration of applications on the desktop, and describe the organization of the desktop in terms of accessing the applications within a system.

7.4 SYSTEM SUPPORT

7.4.1 System Support Functions

Each system should include the resources and utilities required to support the overall tactical functionality provided by the system. In addition, each system should provide performance support modules such as help, online documentation, tutorials, and computer-based training (see section 9.1). System-level resources should allow users to end a session, print screens, review system status, define user preferences, manage alerts (e.g., set alert criteria, review alert logs), change a password, access peripherals, and perform file management. Utilities or tools available at a system level should include basic features (e.g., calendar, notepad, calculator), COTS software (e.g., word processing, spreadsheet, electronic mail), macros, and briefing presentation support.

Users should have the option of printing the entire screen or a specific window with and without its secondary windows. The system should provide the appropriate security marking (without the classification bar color) and produce an accurate translation of color shading from the display to the hardcopy medium. In addition, users should be able to create, access, modify, and delete files and to do so without detailed knowledge of the file system structure. For example, windows for file manipulation might use list boxes for navigating between directories (so that users do not need to know full path names) and selecting directory and file names (so that users do not have to type them).

Each system should implement a screen saver (i.e., blank the screen if there is no user input for a period of time, e.g., three minutes) and provide users with the ability to rapidly suppress a display (i.e., quickly remove it from view to prevent the compromise of protected or classified data). In addition, each system should allow users to temporarily suspend a session without completely logging out. The system should continue all active processes but should not allow any interaction (i.e., accept any keystrokes) unless a user logs in to continue the session. Finally, if navigation among multiple overlapping windows is required, each system should consider providing users with the ability to quickly identify (e.g., by selecting an Active Windows menu option) all open windows, obtain information about them (e.g., the associated application, date created, size), and move input focus among them.

7.4.2 User-Specified Interface Settings

Each system should identify the set of interface parameters (e.g., functions assigned to pointing device buttons, screen location for displaying icons, criteria for alert generation, printer destination, elapsed time before automatic logout) that users are allowed to set. These parameters may vary depending on the specific system functions to which the user is allowed access. The system should set a default value for each parameter that, unless changed by users, applies to all applications within the system. Some parameters may be in effect for the current session only, and some may be saved and implemented whenever the user logs in to the system. Users should be able to review the parameters and reset them at any time during a session. When users end the session, only the settings in effect for the current session should revert to the default values specified by the system.

7.4.3 System-Level Navigation Aids

Each system should implement navigation aids to support users in quickly accessing specific functionality within the system. These aids can be used to supplement the set of menus available in the system menu bar and provide an alternative representation of system functionality from which users can navigate quickly to a desired application or window. Figure 7-6 presents a graphic overview of the functions available in an Army system; when users select one of the items in the graphic, they are automatically taken to that part of the system, and the appropriate window(s) are displayed on the screen.

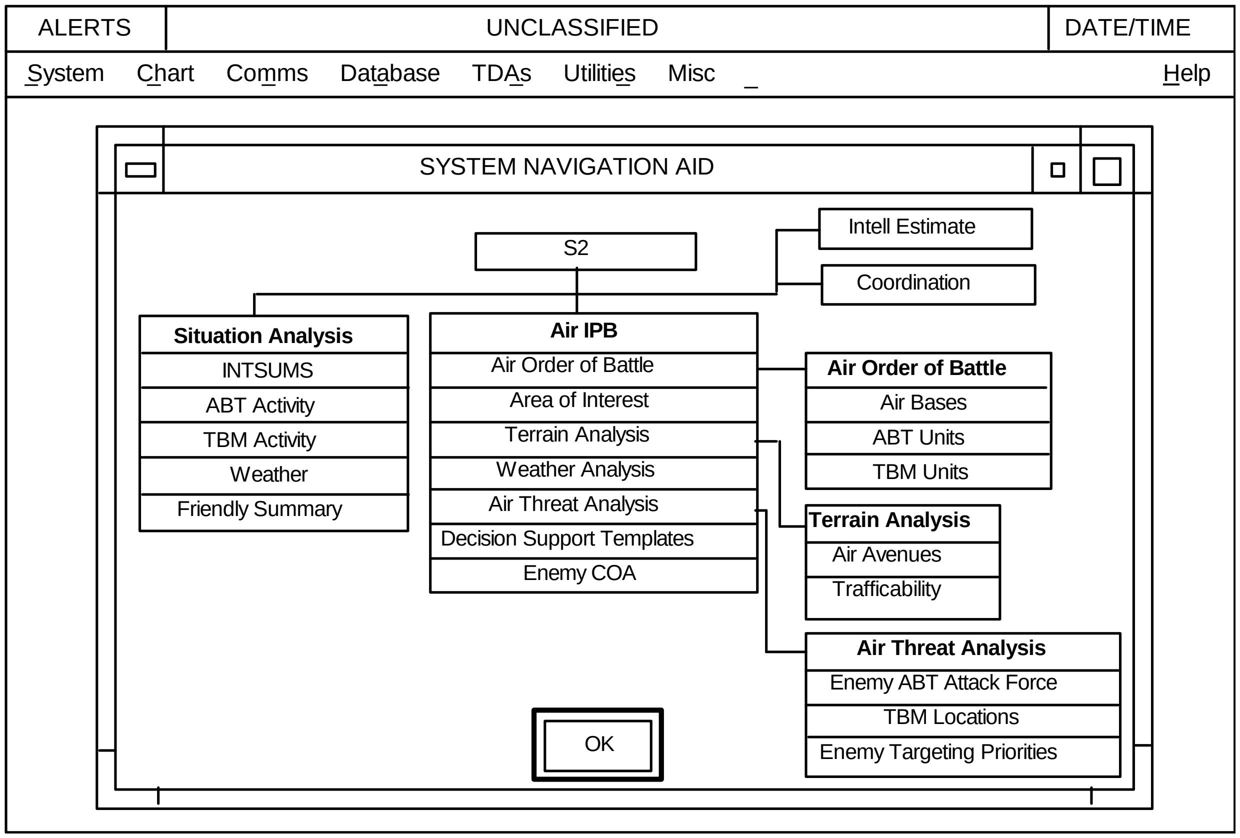


Figure 7-6. Example graphic navigation aid (from the Army guidelines).

Each system should provide navigation aids that allow users to tailor system-level menu contents to the specific task being performed. These aids are particularly important in systems where users have access to a large number of software modules and menus may extend to or beyond the bottom of the screen. These aids should allow users to hide or display groups of menu options for modules with related functionality. These aids should function solely as a “hide/show” toggle that customizes menu contents and should not affect the underlying processes for any of the modules. Navigation aids can be provided as a separate pull-down menu in a system or application menu bar or take the form of a palette displayed along the edge of the screen.

7.5 SYSTEM LOGOUT

7.5.1 User-Initiated Logout

To end a session, users should select a Logout option from the appropriate menu in the system menu bar. Users shall be prompted to confirm the action if any unsaved data have been generated and, if necessary, to save the data. Where appropriate, users should be prompted to log out of any applications to which they were required to log in. During the logout process, all processing in application windows shall stop, and all windows (including the system window) shall be removed from the screen. When logout is complete, the initial login window shall be displayed.

7.5.2 Automatic Logout

If a system implements automatic logout, a standard time (e.g., 15 minutes of user inactivity) should be designated by the system before automatic logout is initiated. This period should be modifiable by users as a user-defined parameter. The system should display a message during periods of inactivity indicating the action (e.g., keystroke, pointing device movement) needed to avoid automatic logout. In addition, an auditory signal should be presented at intervals during the period of inactivity. When automatic logout occurs, any unsaved data should be saved, and a message should be displayed indicating that automatic logout has occurred and providing the file name where any data have been saved.

8.0 APPLICATION-LEVEL WINDOWS

8.1 WINDOW ORGANIZATION

Application windows have four major window areas, shown in figure 8-1: a window title, a window menu bar, a main area, and a message bar.[47] All of the windows in an application shall contain a title and a main area; the inclusion of a window menu bar and a message bar depends on the type of window and the functionality it provides.

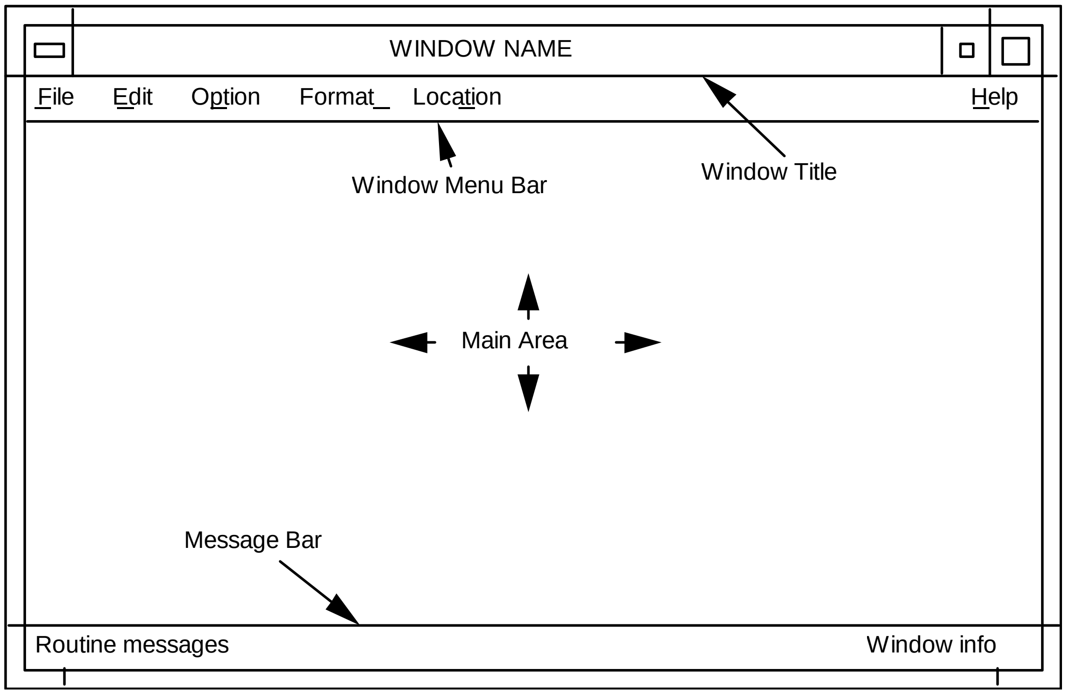


Figure 8-1. Recommended format for application windows (from the JMCIS style guide).

8.1.1 Window Title

A window title shall appear in the title bar of the window. The title shall be centered in the title bar and presented in upper-case letters. Each window title in an application should be unique. If an application contains multiple primary windows, the title should indicate the purpose of the window. The title of a dialog window should also describe its purpose and, if necessary, should include the name of the application so that users can relate the information in the window (e.g., an error message) to the application that generated it. The window title should not contain information such as the version of an application or the full path name for a file. In addition, the window title should not be used to present dynamic information such as messages to the user. If a secondary window is displayed when users select an option in a pull-down menu, the title of the window should match the wording of the option.

8.1.2 Window Menu Bar

A window may include a menu bar containing pull-down menus that provide access to application functions. A menu bar is more likely to be part of the primary window(s) of an application but may also be included in a dialog window. If a window includes a menu bar, it shall appear below the title bar. The menu bar should contain no more than ten menu titles plus Help; no other types of controls shall appear in the menu bar. The menu titles should begin at the left margin of the menu bar and extend rightward, with Help available at the right margin of the menu bar. Commands (e.g., push buttons) should not be included in the menu bar. Each menu title should include a mnemonic to support selection from the keyboard. The mnemonic for the Help menu title should be H; other mnemonics should be selected, as appropriate, from table 5-1 and conform to the guidelines presented in section 5.1.2.5. If one of the menus in the menu bar can be torn off, then this feature should be available for all of the menus in the window.

Except for Help, the options in a window menu bar should not duplicate functionality that is available in the system menu bar. For example, an application that performs a what-if analysis on contact data on a map display should access map and plot control from map display menus and not duplicate this functionality in an application menu bar. In addition, the title of each application menu should be unique and not duplicate the titles that appear in the system menu bar. Finally, if the same menu appears in different menu bars within the application, the menu should have the same title and contain the same options (worded the same and in the same order) wherever it appears.

Developers should adhere to Motif conventions concerning menu design and content to the extent possible, given that operational requirements frequently demand unique functionality that is not defined in the Motif Style Guide. Whenever possible, the standard Motif menus should be used so that users are provided with a common and consistent means for accessing generic functions across applications. If developers create their own menu structure, they shall do so in accordance with the specifications presented in this style guide.

The first menu in an application menu bar should contain options that enable users to work with the data in the window as a whole. This menu should be titled File or an application-specific term with comparable meaning and should, at a minimum, include an option to exit the application. If the menu includes any of the following options, they should be ordered: New, Open, Save, Save As, Print, Close, Exit. If an Edit menu is included in an application menu bar, it should contain options that enable users to modify the data in the window. If the menu includes any of the following options, they should be ordered: Undo, Cut, Copy, Paste, Delete, Select All, Deselect All. The Help menu should provide access to additional information about the content of the window. The contents of this menu are described in section 9.1.

8.1.3 Main Area

8.1.3.1 Arrangement of Window Controls

The main area of a window should be used to present text and/or graphic information and organized into subareas based on the nature of information being presented or the type of action required by users. Controls that perform similar or related functions should be arranged in sets and presented together in a control panel in a window, as shown in figure 8-2. The panel should be surrounded by a frame and clearly labeled to indicate the function performed by the controls presented in the panel. If a control panel includes a title, it should be presented in upper and lower case letters and should not be followed by a colon. The controls should be grouped into one or more rows or columns as appropriate to the function performed by the controls. For example, radio buttons or check buttons in a panel should be arranged vertically so users can scan them easily and quickly.

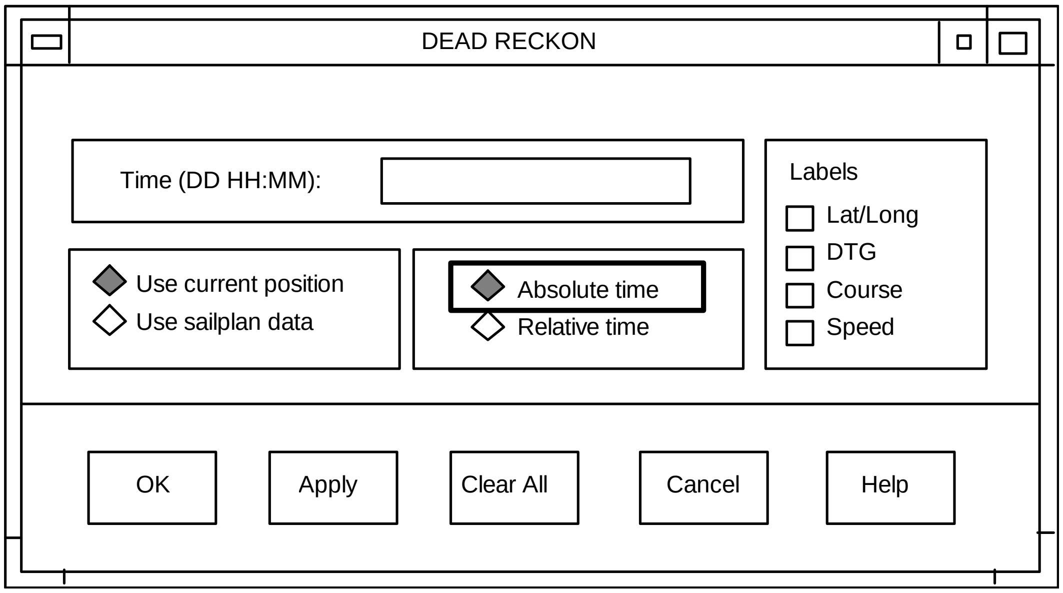


Figure 8-2. Example of grouping objects into control panels (from the JMCIS style guide).

Controls that refer to similar kinds of options should appear together in a set, and controls that refer to different kinds of options should be grouped separately. If scroll bars are needed, they shall be located to the right (for vertical scrolling) or at the bottom (for horizontal scrolling) of the area to be scrolled. When scroll bars are provided, they should scroll only the main area of the window and not the menu bar or message bar in the window.

When a window is displayed, all of the controls within the window should reflect the current state of the application. For example, a dialog window that allows users to change the font size of text displayed by the application should be displayed with the current font selected. If the window is modeless, any changes to the application should be updated in the window while it is displayed. If there is a preferred or expected choice within a window, a default option may be defined within a control (e.g., a list box) or group of related controls (e.g., a set of check buttons), and this choice should be selected (i.e., highlighted) when the window is initially displayed. If the expected choice cannot be anticipated, then a default should not be designated when the window is displayed.

When a window containing a group of check buttons is first displayed, the default state of the check buttons should be deselected (rather than selected) so that the user's task in the window is to select the settings they want to execute, rather than to deselect the settings they do not want to execute. If certain controls become unavailable, they should be dimmed and not available for selection. If certain controls are never available to users (e.g., if access to them is password controlled), they should not appear in the window.

8.1.3.2 Use of Borders and Frames

Lines, borders, and frames should be used to visually group related controls and separate them from unrelated controls within a window. In most cases, a thin, single-line border should surround a group of controls. Variations in line thickness, width, and height should be minimized; no more than three line weights should be used at one time, and lines should be consistent in height and length. In addition, developers should be careful not to overuse frames and borders within a window; for example, if a border is placed around a group of controls, another border surrounding multiple groups of controls should be used only if it is needed to support window functionality and it does not result in unnecessary clutter within the window.

The heading or title for the group of controls should be placed either inside the border or in the border. The heading should be either left justified or centered within the frame. If the heading is longer than the text in the controls, the size of the frame should be extended so that it is wider than the heading.

8.1.3.3 Arrangement of Push Buttons

The bottom of the main area should be reserved for push buttons indicating the actions that can be taken in the window. The push buttons should be displayed horizontally and centered at the bottom of the window. They should be ordered from left to right based on the sequence in which they will be used, with the most frequently used button on the left. Buttons indicating positive actions should be toward the left, followed by buttons indicating negative actions and canceling actions. If a default push button is included in a window, it should be the leftmost button. A Help button should be included in every window and should be the rightmost button. Push buttons should appear in the same order throughout the application.

If users can perform multiple actions within a window and these actions can affect different objects or data elements in the window, push buttons should be labeled to reflect the object(s) that each push button affects. In addition, push buttons should be arranged in the window so that they are located near the object(s) to which they relate. Push buttons related to overall window functionality (e.g., OK, Cancel) should be placed along the bottom of the window.

If users can perform mutually exclusive actions (e.g., Pause and Resume) in a window, the window should contain a push button for each action, and the label of the button that is not available at the time should be grayed out. The window should not contain a single push button whose label changes to indicate the action available. Likewise, a window should include push buttons that allow users to perform all of the actions associated with the functionality provided by the window. For example, if a window allows users to define the parameters for an overlay and then draw it on a map, the window should also include a push button for users to be able to remove the overlay from the map. Every window should provide a push button (e.g., Close, Cancel, Exit) that allows users to dismiss the window. This push button should be in addition to a Close option in the Window Menu for the window. Close and Cancel should not be included as push buttons in the same window.

The push button actions used in a dialog window depend on the mode of the window. Modal windows usually contain the following push buttons (in the order indicated):

OK, Cancel, Help.

Modeless windows usually contain the following push buttons (in the order indicated):

OK, Apply, Cancel, Help
OK, Apply, Reset, Cancel, Help.

The OK button should be the default in modal windows and in modeless windows that are transient (i.e., displayed for only a single action). Apply should be the default in modeless windows that are likely to be displayed for multiple actions.

8.1.3.4 Palettes and Icon Bars

Developers should provide a palette when users interact frequently with objects and actions in a window. A palette usually consists of a group of buttons that define a set of related exclusive modal choices (e.g., a set of drawing tools). The choices in the palette should be presented as icons rather than text whenever possible and arranged vertically at the left or right edge of a window, as shown in figure 8-3. An icon bar is a form of palette that includes a group of buttons for applying settings (e.g., for selecting font style or size) and executing actions (e.g., for printing), arrayed horizontally across the top of a window, directly below the menu bar. If a palette or an icon bar is included in a window, it should provide redundant access to actions available elsewhere (e.g., in a pull-down menu) in the window.

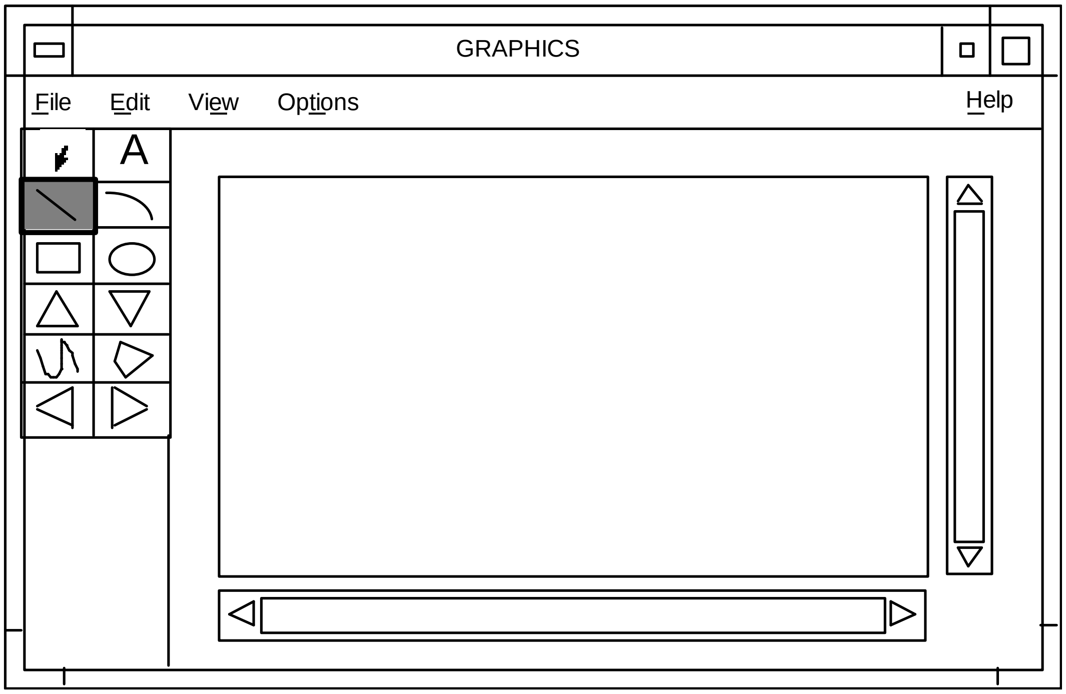


Figure 8-3. Recommended format for action icons in windows (from the JMCIS style guide).

A palette or an icon bar should contain no more than 20 buttons of equal size. The contents of each button should be large enough to illustrate the alternative and permit easy selection with the pointer. If the contents are presented as icons, their meaning should be consistent across windows within an application as well as between applications within a system. The Bellcore style guide recommends that the normal appearance of buttons in a palette is “flat” (i.e., not raised or recessed) and the select state is indicated by highlighting (i.e., change in color) and a recessed appearance.

The buttons in a palette should be positioned so that their edges are nearly touching (allowing enough space for the location cursor to be displayed). The buttons should be arranged in an order expected by users; if one does not exist, then they should be arranged by frequency of occurrence, sequence of use, or importance. A palette should be movable so that users can reposition it as needed. Users should also be able to navigate within a palette from the keyboard and to remove the palette from the window.

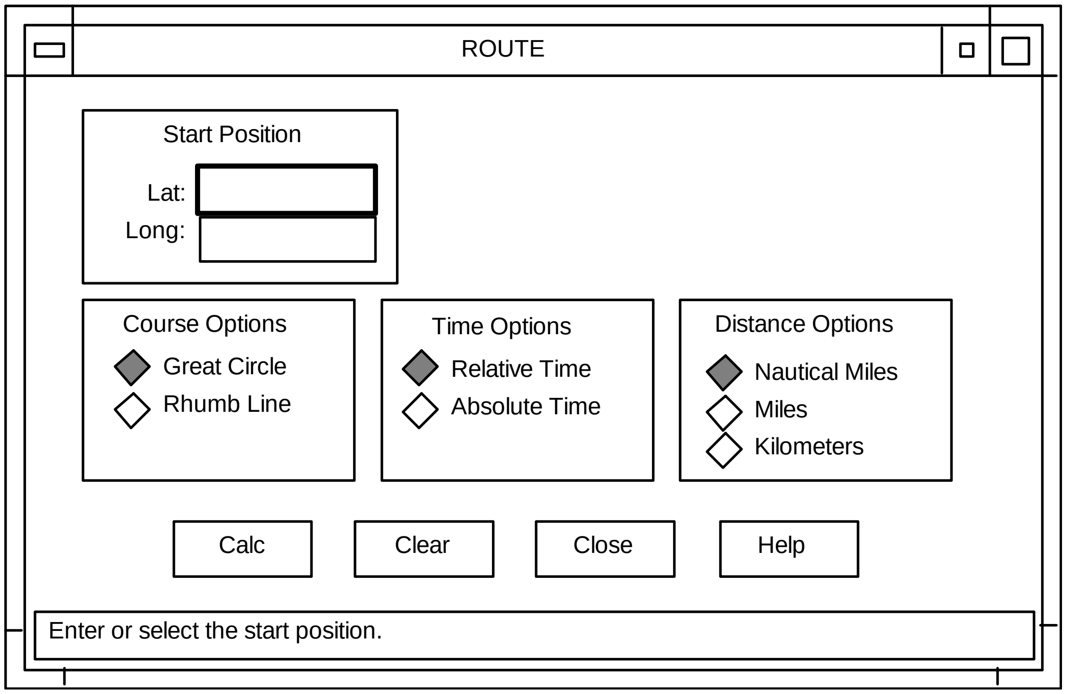
When users select one of the alternatives in a palette, its appearance should change to indicate its selected state, and the appearance of any other choices that became unavailable should change to indicate that they cannot be selected. The alternative should remain selected (i.e., highlighted) as long as the mode invoked by the alternative remains in effect. In addition, the pointer shape should change to indicate the type of operation users can perform while in the mode. The pointer should retain this shape whenever it is in the window to which the mode applies; if users move the pointer outside this window, the pointer should change to the appropriate shape. Developers should design the palette so that either it provides an alternative that returns to a neutral state (i.e., disables the mode) when selected, or users are automatically returned to a neutral state when they execute an action while in the mode.

8.1.3.5 Visual Design Considerations

Developers are encouraged to follow recommendations provided by Kobara in Visual Design with OSF/Motif regarding the construction of widgets so that window components are sized, spaced, and positioned in accordance with Motif design philosophy. These recommendations are included with the resource settings for Motif widgets presented in table A-2 of appendix A.

8.1.4 Message Bar

A message bar should be included along the bottom margin of a window to provide noncritical application messages to users or to indicate when the actions to be executed in a window differ from what is normally expected. The left side of the bar should be used to present routine messages to users, provide simple help (e.g., actions needed to complete execution of a command), and indicate status pertaining to the window (e.g., feedback such as “Drawing map”). The right side of the bar should be used to identify information about the window (e.g., the name of the map or document, the page number such as "3 of 6"). The message area should display read-only text; users should be unable to type or modify any information presented in this area. A message bar should be part of the primary window(s) of an application but may also be included in a dialog window (as in figure 8-4). The message bar may be an area that is set aside in each window for this purpose, or may be an area that is used temporarily to present a message and then returned to its previous use.



Enter or select the start position.

Figure 8-4. Example message bar in a dialog window (from the JMCIS style guide).

8.2 DIALOG WINDOWS

8.2.1 Prompt Windows

Applications should display a prompt window, shown in figure 8-5, to request information that is needed to continue processing. When a prompt window is displayed, processing by the application is usually suspended until users respond to the window. A prompt window should include a title, a message indicating the information requested, a text field for typing the information, and the following push buttons (in the order indicated):

OK, Cancel, Help
OK, Apply, Cancel, Help
OK, Apply, Reset, Cancel, Help

The title of the prompt window should indicate the process or application that generated the prompt, and the message itself should be phrased in language that is meaningful to users.

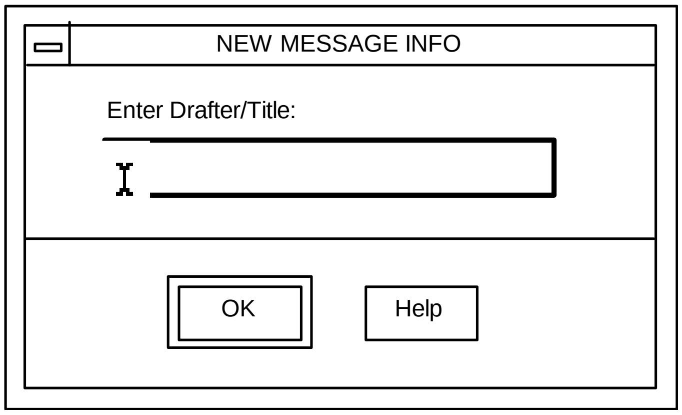


Figure 8-5. Example prompt window (from the JMCIS style guide).

8.2.2 Message Windows

8.2.2.1 General Design Guidelines

When an application displays a message window, it should be as small as possible, yet large enough to display the information required. A message window should be uniquely identifiable by its appearance and be positioned at a standard location on the screen whenever it appears. Users should be able to move a message window but not minimize or resize it.

When a message window containing critical information is displayed, it should be accompanied by auditory feedback (e.g., a beep) as a secondary indicator to attract the user's attention. Users should be able to set auditory signals at a very low intensity or disable them as required by operational conditions (e.g., rig-for-quiet on submarines). If a process running in a minimized window generates a message containing routine information, the appearance of the window icon should change to indicate the presence of the message. If the message contains critical information about the process, a message window with the information should be displayed directly on the screen, in front of whatever windows are currently displayed.

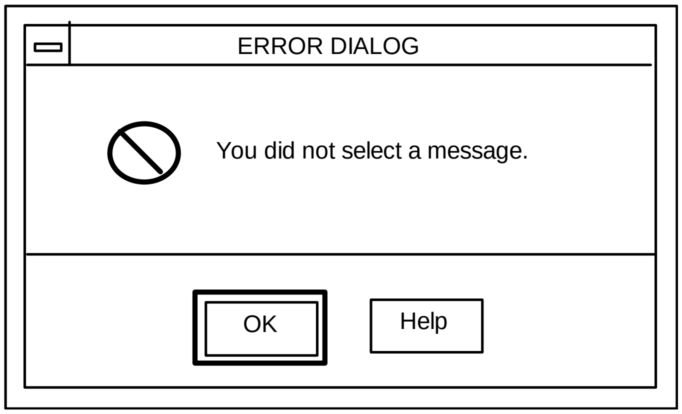
A message window should include a title, a symbol indicating the message type, the message itself, and one or more push buttons. As in prompt windows, the title of a message window should indicate the process or application that generated the message since multiple applications may be running simultaneously and users should be able to identify the source of the message (e.g., the message may be associated with an application that is running in a minimized window).

The message itself should use language that is meaningful to users and require no further documentation or translation. The text in a message should be left justified within the window. When a message contains more than one sentence, the important information should be placed at the start of the message. The text should be worded so that the action users are asked to perform can appear as a push button in the window. For example, a window displaying the message "Confirm deletion of file" should contain Delete and Cancel push buttons.

A message window should include at least one push button, presented along the bottom of the window, that solicits a response from users. Whenever possible, action verbs, rather than Yes and No, should be used in message windows to reduce the likelihood of users having to execute the same action by selecting Yes in one application but No in another.

8.2.2.2 Error Message Windows

Applications should display an error message window, shown in figure 8-6, to inform users of an error that has occurred. Error messages should focus on the procedure for correcting the error, not the action that caused the error. If the error is repeated, the focus of error messages should change to include recommendations for preventive actions. The application should suspend processing until users respond to the window. The window should contain an error symbol, a message, and the following push buttons (in the order indicated):
OK, Help.


ERROR DIALOG

You did not select a message.

Figure 8-6. Example error message window (from the JMCIS style guide).

8.2.2.3 Information Message Windows

Applications should display an information message window, shown in figure 8-7, to convey noncritical information that requires acknowledgment by users. (The message bar of a window should be used for frequently presented messages that require no acknowledgment by users.) An information message window shall not interrupt processing by the application. Applications should not present timed-information windows (i.e., message windows that present information for a fixed time period) and then resume processing without requiring a user response. The information message window should contain an information symbol, a message, and the following push buttons (in the order indicated):
OK, Help.

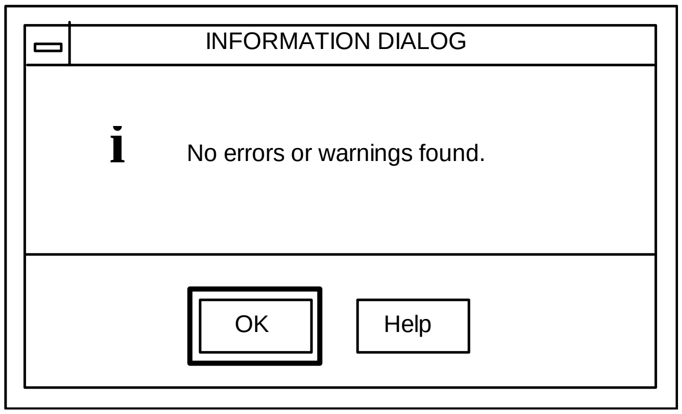


Figure 8-7. Example information message window (from the JMCIS style guide).

8.2.2.4 Question Message Windows

Applications should display a question message window, shown in figure 8-8, to request clarification of a previous response. The application should suspend processing until users respond to the window. The window should contain a question symbol, a message, and the following push buttons (in the order indicated):

Yes, No, Help
Yes, No, Cancel, Help.

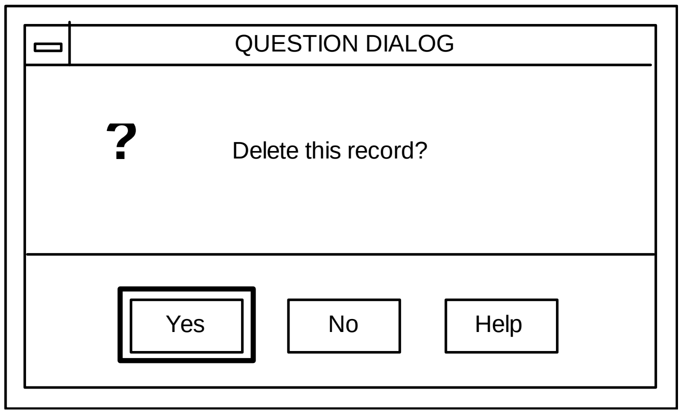


Figure 8-8. Example question message window (from the JMCIS style guide).

8.2.2.5 Warning Message Windows

Applications should display a warning message window, shown in figure 8-9, to present critical messages concerning the consequences of an action. A warning message window should be displayed when users attempt to execute a destructive action. An audio signal should accompany the window to alert users to the warning (see section 8.5.5). Processing by the application should be suspended until users respond to the window. The window should contain a warning symbol, a message, and the following push buttons (in the order indicated):

Yes, No, Help
OK, Cancel, Help.

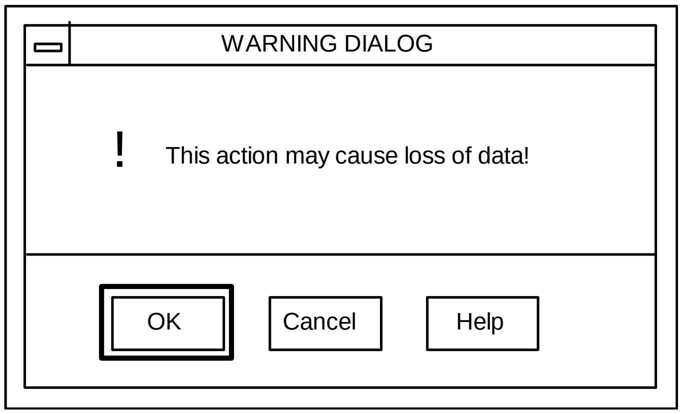


Figure 8-9. Example warning message window (from the JMCIS style guide).

8.2.2.6 Working Message Windows

Applications should display a working message window, shown in figure 8-10, when the processing time in response to a user's request exceeds five seconds or when a user may want to cancel the operation that is in progress. The window should not interrupt processing by the application. The window should remain displayed until the action is complete or until the window doing the processing is minimized. When processing is complete, the window should disappear. Users should be able to cancel the operation in progress if desired, and they should have to confirm the action before it is executed if unsaved data will be lost.

The window should contain a working symbol, a message, and the following push buttons (in the order indicated):

OK, Help
OK, Cancel, Help
OK, Stop, Help
OK, Pause, Resume, Stop, Help.

If the processing time will be lengthy, the window should be updated to indicate the status of processing (e.g., a message stating "5% sorted") or include a scale showing the percent executed or transferred.

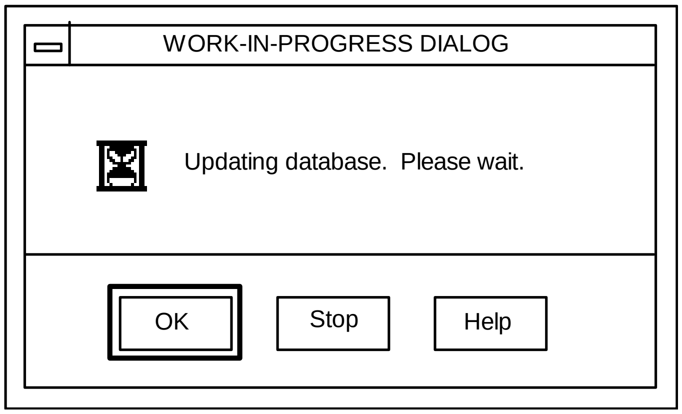


Figure 8-10. Example working message window (from the JMCIS style guide).

8.2.3 Command Windows

A command window allows users to issue new commands by typing them and to review and reissue previous commands. A command window should contain a list box that displays a command history and a text field for entering commands; the window does not include any push buttons (see figure 8-11). The list should include a vertical scroll bar when the command history exceeds the visible area in the list. The text field should be wide enough (maximum length 60 characters) so that users can view and read an entire command; a horizontal scroll bar should not be included unless command lines are unusually long. A command window shall be modeless.

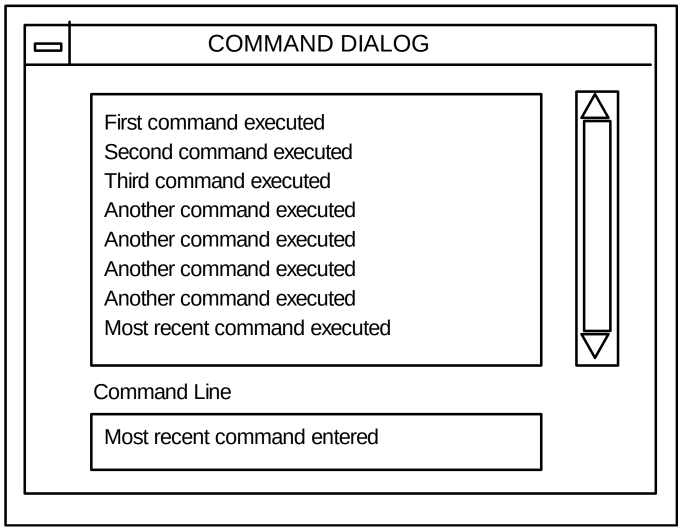


Figure 8-11. Example command window (from the JMCIS style guide).

Users can either type a command in the text field or select an item from the list to display in the text field. When they press <Return>, the command should be executed and added to the bottom of the command history list. <Tab> should move the location cursor between the list and the text field; keyboard navigation available in a list box (see section 6.5.2) should also be available in the command history list.

8.2.4 Selection Windows

A selection window, shown in figure 8-12, allows users to make a selection from a set of choices presented in the window. The window should include a list box containing the choices available, a text field for displaying and editing the choice, and the following push buttons (in the order indicated):

OK, Cancel, Help
OK, Apply, Cancel, Help.

The list should provide a vertical scroll bar when the number of items exceeds the visible area in the list. Users can either type an item in the text field or select it from the list to display in the text field. If users type in the text field, the list should scroll to that item in the list. If the text does not match any of the items in the list, users should be prompted as to whether they want to add the item to the list. When they select OK or press <Return>, the selection should be executed and the window closed.

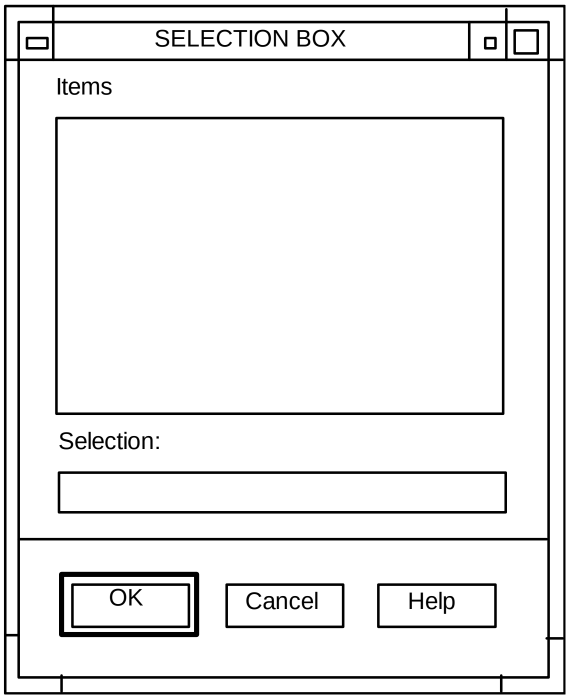


Figure 8-12. Example selection window (from the JMCIS style guide).

A file selection window is a specialized selection window that allows users to find and select a file for processing. The window should include (1) a text field for displaying and editing a directory filter used to select the files to be displayed, (2) a list box for displaying file names, (3) a list box for displaying subdirectories, (4) a text field for displaying and editing a file name, and (5) OK, Filter, Cancel, and Help push buttons. Users should be able to select a directory either by typing in the Filter text field or by selecting one of the items in the Directories list. Similarly, users should be able to select a file either by typing in the Selection text field or by selecting one of the items in the Files list.

8.3 COLOR USE IN WINDOWS

8.3.1 Application Windows, Menus, and Controls

The color set for application window components, menus, and controls is presented in table 8-1 and figure 8-13; Motif resource settings (and RGB values) are provided in Appendix A.[48] Application developers shall design their software to support this color set. In addition, in order to accommodate the addition of new color sets to the specifications, developers shall design application software so that Motif color-related resources reside in a single file and are not hardcoded into the application.

Note to reviewers: Multiple color sets for the range of operational environments in which GCCS-based systems will be used need to be defined, with selection of a color set available as a user preference setting.

[48]. Appendix A also includes resource settings for the Motif version of software modules (e.g., applications taken from release 2.1 of Unified Build) originally developed with the Wizard toolkit.

Table 8-1. Appearance of objects in application windows.

Object	Appearance
Window Components	
Window Frame	
Active	TBD background, TBD text
Inactive	TBD background, TBD text
Window Menu	TBD background, TBD text
Client Area	TBD background, TBD text, TBD location cursor
Window Icon_	
Active	TBD background, TBD text/graphics
Inactive	TBD background, TBD text/graphics
Message Window, Command Window, Selection Window, File Selection Window	TBD background, TBD text
Menus	
Window Menu Bar	TBD background, TBD text
Pull-down Menu, Pop-up Menu, Option Menu	TBD background, TBD text
Controls	
Push Button	TBD background, TBD text/graphics, TBD when selected
Radio Button, Check Button	TBD background, TBD text, TBD when selected
Text Field	
Editable text field	TBD background, TBD text
Error in text entry	TBD background, TBD text
Noneditable text field	TBD background, TBD text
Static text (i.e., label)	TBD background, TBD text
List Box	TBD background, TBD text
Two-state coding	TBD background, TBD and TBD text[49]
Tactical coding	TBD background, text color as indicated in section 8.3.2
Scroll Bar[50]	TBD trough

[49]. Although Version 1.2 of Motif allows only one foreground color in list boxes, multiple text colors are needed in order to color code tactical information presented in lists; version 2.0 will provide the ability to define multiple foreground colors.

Scale	TBD background, TBD text, TBD trough

—

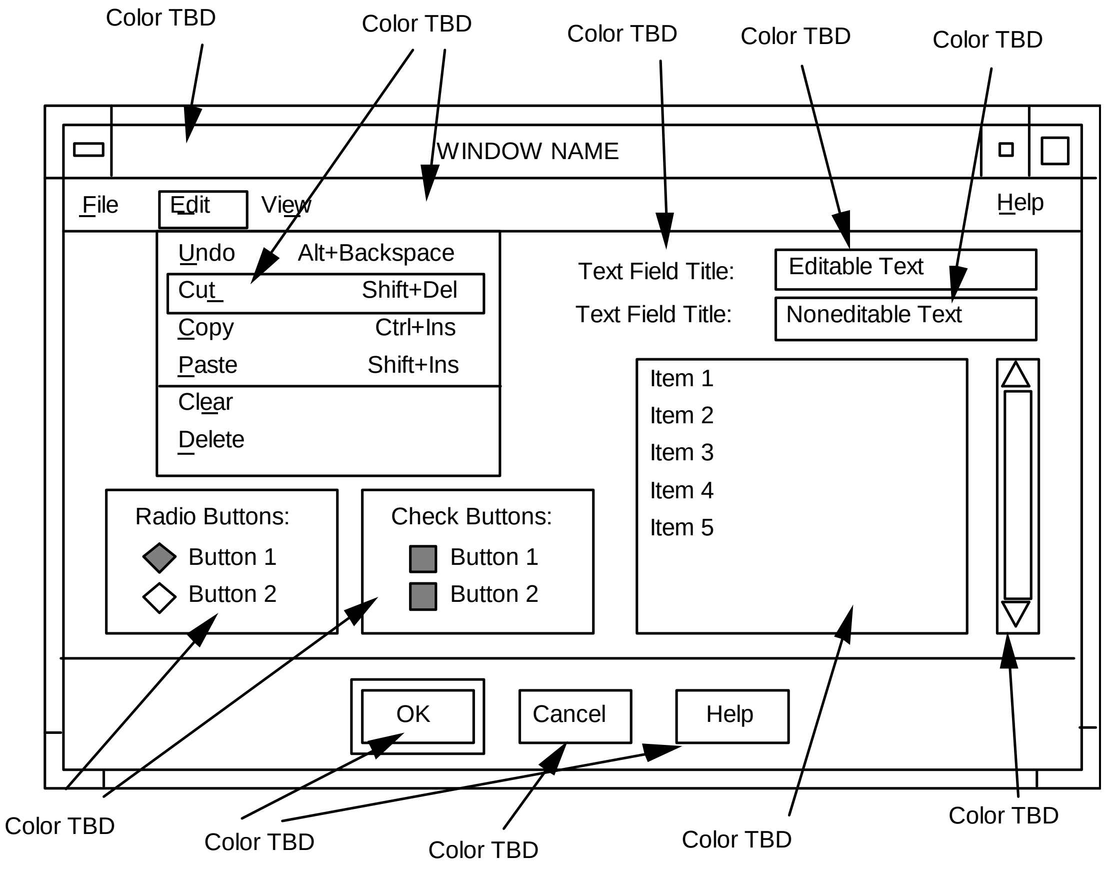


Figure 8-13. Example colors for application windows.

If necessary, developers may modify the object appearance defined here in order to provide the functionality called for by the application; developers doing so should follow the guidelines provided in section 6.8 on standard and nonstandard controls, section 8.3.2 on color coding of tactical information, and section 8.3.3 on color selection. In addition, if necessary to the functionality of the application, users should have the option to select alternative color sets as a user preference setting. If this option is available, it should allow users to change colors for

[50]. The slider and stepper arrows should be the same color as the background of the object to which the scroll bar is attached (e.g., the slider and stepper arrows should be Black if the scroll bar is attached to a scrollable area that has a Black background).

aspects of an application that do not involve coding or status. However, applications shall not be able to alter the set of classification colors associated with each security level.

8.3.2 Color Coding of Tactical Information

Color should be used consistently throughout an application, in accordance with published standards (e.g., MIL-STD 2525 for threat color coding), and in ways that match user expectations. Color (other than that specified in table 8-1) should be used in the main area of a window only when necessary and primarily as a coding scheme to highlight or redundantly code critical elements in the window. The arbitrary use of many colors may reduce the likelihood that significant color coding will be interpreted quickly and accurately by users. Color should be used consistently within an application and, to the extent possible, between applications within a system.

If color is used to impart tactical meaning, it should be used as a redundant code with another display feature (e.g., shape, symbology, content of text) and not as the sole basis for coding. If color coding is used, each color should represent one category of displayed data. Color should be used consistently throughout an application and in ways that match user expectations. Applications that use color to indicate threat or system status should adhere to the following conventions:

Threat Status	System Status
Cyan = Friendly	Green/Blue = Operational/Normal/Noncritical
Red = Hostile	Yellow = Caution/Questionable
Green = Neutral	Red = Inoperative/Error
Yellow = Unknown	
Purple = Ambiguity	

The colors selected to convey tactical meaning should be the same throughout an application, and their use should be restricted to that function only. If one of these colors is assigned another meaning, a different shade should be selected so as to minimize the likelihood of confusion with the convention. Mayhew in Principles and Guidelines in Software User Interface Design provides the following ISO guidelines regarding color coding:

To denote:	Use:
Larger size:	Saturated or bright colors
Smaller size:	Desaturated or dark colors
Equal size:	Colors equal in brightness
Heaviness:	Saturated, dark colors
Lightness:	Desaturated, light colors
Depth:	Saturated, dark colors
Closeness:	Saturated, bright colors
Height:	Desaturated, light colors
Low-end continuum:	Short-wavelength dark colors
High-end continuum:	Long-wavelength bright colors

Color can also be applied to tactical information for the purpose of alerting. In this case, only the information to which the application wants to direct user attention is assigned a unique color, with all other information displayed according to the color specifications defined for the application. As in color coding, a standard meaning in terms of alert criticality is assigned to each color, and that color is used only to convey this meaning. While alerting is usually indicated by assigning color to text information (e.g., in a list or table), colored icons can also be defined and appended to the information. Applications that use color for alerting should adhere to coding conventions defined here. Examples of how these colors might be applied as alert indicators are indicated below:

Vulnerability Time
Red = Vulnerable now
Yellow = Vulnerable in XX minutes
Green = Vulnerable in YY minutes (where YY is greater than XX)
Blue = Not vulnerable

Probability of Detection
Red = High probability (greater than XX percent)
Yellow = Medium probability (less than XX percent but greater than YY percent)
Green = Low probability (less than YY percent)

Confidence Factor
Red = Unknown
Yellow = Low
Green = High

Priority
Red = High
Yellow = Medium
Green = Low

Probability of Hostile Action
Red = Imminent
Yellow = Probable
Green = Possible
Blue = None

Action Items
Red = Now
Yellow = In XX minutes
Green = No time limit
Blue = No action required

A background color should be selected that provides sufficient contrast with all of the colors used for coding so that all of the text and graphics in a window is clearly visible and discriminable by users. For example, a gray background (e.g., gray10 for water and gray40 for land) is recommended when displaying tactical symbology in a map window.

Applications that use color to distinguish among states or categories of related information (e.g., to code the items in a list according to active vs. inactive processes, user-entered vs. system-generated data) should designate specific colors (e.g., White and Khaki1) for this type of coding and then use these colors whenever these types of distinctions need to be made. Applications should minimize arbitrary variations in color since they may confuse users by providing misleading, inappropriate, or inaccurate cues regarding distinctions in information being presented.

8.3.3 Guidelines on Color Selection

When developers use color in an application, they should do so according to the following general guidelines:[51]

a. Heavy use of highly saturated colors, opposing colors (e.g., yellow and blue), and colors at spectral extremes (e.g., yellow and purple) should be avoided because they may cause afterimages, shadows, and depth effects.

b. Pure white text should not be displayed on a pure black background because this combination produces halation which makes the text less readable. In addition, because of focusing problems with blue, this color should not be used in text or for any critical information.

c. If color is used to distinguish message windows, it should appear in the message symbol (e.g., a warning hand in the warning message window) and not be used for message text.

d. The number of colors used to code the information in an alphanumeric display should not exceed seven, and only four codes should be displayed at any one time. The number of colors used to code information on graphical displays should not exceed eight or nine, since users have difficulty discriminating among more than this number of colors. When information in a display is color coded, the meaning of the code should be defined in a legend at the bottom of the window (e.g., in the message bar).

e. Slight shade changes in color should not be used to show gradation or choice. Shade differences are usually difficult to see, especially on a varied background such as a map. While normally discouraged as a graphical area discrimination technique, color shading, if used, should be of sufficiently differing intensity as to be obvious and should not be used to determine object selection or for control of the application.

f. The background color behind text should not be changed to show a change in system status because changing the background color usually reduces the readability of the text. Instead, the change should be signaled by changing the color of an object (e.g, a box or circle) next to the text.

g. Both brightness and type of lighting (e.g., incandescent vs. fluorescent) can affect how colors are perceived. For example, bright ambient light desaturates display colors, leading to degraded color identification and discrimination.

h. At normal viewing distance from a screen, maximal color sensitivity is not reached until the size of a colored area exceeds about a three-inch square. Smaller size images become desaturated and change slightly in color. Also, small differences in actual color may not be discernible, and small adjacent colored images may be perceived to merge or mix.

i. Color discrimination is better when color images are displayed on an achromatic background (black, gray, or white) and achromatic images are displayed on a color background. If color images are displayed on color backgrounds, then background and symbol colors should contrast in both brightness and hue to ensure legibility.

8.4 TEXT IN WINDOWS

8.4.1 Text Font, Size, and Readability

[51]. More detailed guidelines on color use can be found in the DoD style guide.

The text colors indicated in table 8-1 should be used in windows; if a different text color is used, it should have sufficient contrast with the object or window to ensure good readability.[52]

Developers should use fixed width, sans serif fonts when presenting text in windows.[53] A heavier (two-pixel stroke thickness) font should be selected so that text is readable both when presented normally and when dimmed or dithered (e.g., to show that an object is unavailable). Fixed (rather than proportional) width fonts are recommended to support ease of scanning (e.g., columnar data), and sans serif (rather than serif) fonts are preferred for ease of readability, especially if screen resolution is degraded. While proportional-width fonts make efficient use of space within a window, they should not be used in text entry areas because the text cursor can be difficult to see when placed between characters displayed in these fonts. Developers may select different fonts within a window to reinforce distinctions between static information (e.g., titles, labels) and system- or application-generated data (e.g., default values in a text field). Applications that provide word processing capabilities should provide a choice of text fonts as a user-selectable option within the application.

The text in each window should be of sufficient size and thickness to be readable by users when they are seated at the normal viewing distance from the screen. The DoD style guide indicates that at a minimum, character height should be 1/200 of viewing distance; for example, a viewing distance of 36 inches requires a .18 inch character height. Character width should be 50-100 percent of character height, and character stroke width minimum should be 10-12.5 percent of character height. The DoD style guide recommends that characters contain a minimum 7 x 9 dot matrix construction for better readability. Maximum text size should not exceed 10 percent of the available vertical display area of a full-size screen.

If the text is part of a window that is likely to be used in a briefing presentation, the text should be readable at the normal audience viewing distance when the text is projected on a large-screen display. The DoD style guide recommends minimum character sizes of 10 x 14 dot matrix format and the use of double stroke widths and negative contrast (i.e., black characters presented on a white background). Applications with a range of viewing requirements should provide text font size as a user-selectable option within the application.

8.4.2 Capitalization

[52]. Minimum luminance contrast for text information will be specified in a future version of the specifications.

[53]. Kobara recommends the use of sans serif, variable pitch fonts (e.g., Helvetica) for labels and system messages and serif, fixed pitch fonts (e.g., Courier) for text entry areas. The specifications presented here recommend the use of sans serif fonts for all text displayed in application windows because these fonts are easier to read under degraded viewing conditions.

Titles and major headings should be presented in upper-case letters; all other text (e.g., labels, directions, messages) should be presented in a combination of upper- and lower-case letters, following standard capitalization rules. A consistent approach to capitalization should be used so that the possibility of textual distraction for users is minimized. All upper-case letters should be used in text only for acronyms and abbreviations and for emphasis. Arabic rather than Roman numerals should be used when the information in a window has to be numbered.

8.4.3 Acronyms and Abbreviations

Acronyms and abbreviations should be used only when they are significantly shorter than the full word and they are commonly understood by users (e.g., are related to normal language or are specific job-related terminology). Abbreviations should be the shortest possible that will ensure uniqueness. Abbreviations should be meaningful and recognizable by users and should be used consistently within an application. Words not commonly abbreviated should not be abbreviated. The DoD style guide recommends that acronyms and abbreviations comply with the following documents:

AR 310-50. Authorized Abbreviations and Brevity Codes

MIL-STD-12D. Abbreviations for Use on Drawings, Specifications, Standards, and in Technical Documents

MIL-STD-411E. Aircrew Station Alerting Systems

MIL-STD-783D. Legends for Use in Aircrew Stations and on Airborne Equipment

New acronyms should be generated according to rules contained in MIL-STD-12D. When abbreviations or acronyms are used, a dictionary should be available to users (e.g., in Help).

8.4.4 Static Text in Windows

Applications should use consistent grammatical structure for all static text (i.e., labels such as titles, headings, and directions) displayed in windows. The wording should contain familiar terms and use task-oriented language of users. Blocks of text should be broken into smaller, meaningful groups, as shown in figure 8-14. If continuous text is being presented, the maximum line length should be 40-60 characters; line lengths of less than 26 characters should be avoided. Paragraphs should be kept short and separated by at least one blank line.

Do this:	Not this:
3 Bn Status o Located at 32UNA100100 o Moving to Contact o 80% Strength o Supported by 9th armor platoon	The 3rd Bn is currently located at the 32UNA100100, moving to contact in sector 8, with 80% strength, supported by the 9th armor platoon.

Figure 8-14. Example of breaking information into smaller text groups (from the DoD style guide).

If continuous text (in the form of complete sentences) is used in windows, it should be phrased in simple sentences, in the affirmative (rather than negative) and in active (rather than passive) voice, as shown in figure 8-15. A sequence of events or steps should be presented in the order they are performed. The referent for “it” or “they” in a sentence should be easily identified. Normal punctuation rules should be followed, and contractions and hyphenation should be avoided.

Do this:	Not this:
Press ENTER to continue.	The user should press ENTER to continue.
Clear screen before entering data.	Do not enter data before clearing the screen.
Select one.	Will you make a selection?

Figure 8-15. Example of wording style (from MIL-HDBK-761A).

8.4.5 Editable Text in Windows

The manner in which editable text (i.e., text entered by users) is displayed in an application window should be appropriate to the task being performed in the window. For example, if users are doing normal word processing, the application should display the text exactly as it is typed by users (i.e., with the same capitalization and punctuation). However, if users are doing word processing to compose a message, the application should display the text in capital letters (even if users type it in lower case) and with the appropriate punctuation. In addition, when presenting stored text in a window (e.g., in a text field), the application should display it in a standard format (e.g., all capital letters), and any text editing performed by users should be converted into this format (e.g., text entered in lower case letters should be converted into upper case) by the application.

8.4.6 Formats for Date/Time and Latitude/Longitude Display[54]

Applications should use the following format when presenting date/time information:

The date should be displayed as YYMMDD, where YY is the last two digits of the year, MM is the month, and DD is the day, or as DD MMM YY, where DD is the day, MMM is the month, and YY is the last two digits of the year.

Time should be displayed as HHMM[SS]Z, where HH is the hour of a 24-hour day, MM is the minute, SS (optional) is the second, and Z is the time zone (Zulu [Z] time is the default).

Date/Time Group (DTG) should be displayed as DDHHMMZ MMM YY, where DD is the day, HH is the hour, MM is the minute, Z is the time zone (Zulu is the default), MMM is the month, and YY is the year.

Applications shall display latitude and longitude in separate fields, with the labels "Lat" and "Long." Latitude should be displayed in one of the following formats:

D{D}H, where D (one or two characters) is the degrees of latitude and H is the hemisphere (N for North, S for South).

DD{MM{SS}}H, where DD is the degrees of latitude, MM is the minutes of latitude (optional), SS is the seconds of latitude (optional, but can only be given if minutes of latitude is given), and H is the hemisphere (N for North, S for South).

Longitude should be displayed in one of the following formats:

D{D{D}}H, where D (one, two, or three characters) is the degrees of longitude and H is the hemisphere (E for East, W for West).

DDD{MM{SS}}H, where DDD is the degrees of longitude, MM is the minutes of longitude (optional), SS is the seconds of longitude (optional, but can only be given if minutes of longitude is given), and H is the hemisphere (E for East, W for West).

8.4.7 Wild Card Characters in Text Searches

An application should provide the capability to enter wild card characters to search for specific text patterns if this capability is appropriate to the functionality of the application.[55] The following wild card conventions should be used when providing this search capability:

[54]. The DoD style guide recommends these formats when presenting date/time and latitude/longitude information.

[55]. The wild card characters included here are taken from the DoD style guide which recommends their use if appropriate to the application.

@ searches for the occurrence of a single upper- or lower-case alphabetic character. For example, abc@d retrieves the strings abcad, abced, and abczd; abc7d and abcddd do not match the search pattern and are not retrieved.

# searches for the occurrence of a single numeric character. For example, 123#4 retrieves the strings 12334, 12394, and 12304; 123x4 and 123554 do not match the search string and are not retrieved.

? searches for the occurrence of a single alphanumeric character (a - z, A - Z, 0 - 9, and punctuation marks). For example, abc?d retrieves the strings abcad, abcAd, abc(d, and abc9d; abcxxd does not match the search string and is not retrieved.

* searches for the occurrence of zero or more alphanumeric characters. For example, abc*d retrieves the strings abcad, abcd, abfklsmd, and abc7d; abcd5 does not match the search string and is not retrieved.

8.5 CONSIDERATIONS IN WINDOW DESIGN

8.5.1 Selecting Controls to Match User Actions

Application developers should select the controls that appear in the window to match the actions that users will execute in the window. The following are guidelines on selecting controls to match user actions:

a. Use primary window(s) for the primary actions within the application and to present controls that are used frequently. Use dialog windows for ancillary application actions and to present controls that are used less frequently.

b. Use radio buttons, an option menu, or a list box when users need to select from discrete values or choices. Use a scale when users need to select from a continuous range of values. Use radio buttons rather than an option menu when it is important for users to see all of the settings available in a group, rather than only the one that is currently selected. If space is limited in a window, use an option menu rather than radio buttons since an option menu takes very little space. In addition, use an option menu to present a set of choices that users select from infrequently. Use a list box when users need to make a choice from a large group of options (more than twelve).

c. Use radio buttons or an option menu when the set of options from which to choose is not likely to change. Options that cannot be chosen should be displayed as unavailable (e.g., dimmed) rather than removed from the set of options. Use a list box when the options from which to choose may change.

d. Use check buttons when users need to choose more than one option from asmall number of options (up to seven) in a group; use a list box otherwise.

e. Use push buttons in a control panel for frequently used selections when space is available for displaying the buttons or when users are already moving the pointing device in the control panel area. Use tear-off menus for frequently used selections if space is limited. Use pop-up menus only when it is critical to the application that users be able to access functions without moving the pointing device. Do not use pop-up menus as the only method for accessing operations because the operations are hidden from view and require users to remember where the operations are provided and how to access them.

f. Use option menus for setting values or choosing from a set of related options and not for selecting commands.

8.5.2 Arranging Information to Match User Actions

Application developers should determine how users will execute the actions called for by a window and then design the window layout so that users can manipulate objects in ways that support task performance. For example, if an application generates information that should be presented a page at a time, the developer should provide users with the ability to page by including push buttons that perform standard paging operations (e.g., Next, Previous). In addition, the objects in a window should be arranged so that users can move quickly and easily among them. The amount of pointer movement and/or the number of keystrokes required to perform a task should be minimized whenever possible. Likewise, objects should be arranged to minimize the amount of hand movement between the keyboard and pointing device when working within a window.

The window layout should be designed according to users' natural scanning order and probable selection sequences. In most cases, this order should be from left to right and top to bottom. For example, when presenting a series of radio or check buttons, the most frequently used button should be the top (or leftmost) button in the group. The window layout should be designed from the perspective of what is logical to users and appropriate to the actions being executed, not what is logical or appealing to application developers.

Applications should provide sufficient blank or empty space around text and graphic objects to support efficient scanning within a window. The DoD style guide indicates that screen density (i.e., the ratio of characters to blank spaces) in text windows should not exceed 60 percent of available character space.

8.5.3 Arranging Information by Importance

The most important information and controls associated with the task being performed in a window should be located in the upper left part of the window, working down to the less important information at the bottom of the window. In addition, the objects in a window should be arranged to accommodate the possibility that users will resize the window. Placing the most important objects in the upper-left corner of the window allows users to manipulate these objects even if the window is reduced to a size that is smaller than normal. Finally, if a window contains task-critical information, the information should be visually set apart from other information in the window so that it can be easily seen by users. The minimum separation should be one character space above and below and two character spaces before and after the critical information.

8.5.4 Designing Windows to Minimize User Memory Load

Windows should be designed to minimize the short-term memory load placed on users when they perform the task called for by the window. The window should contain all of the information relevant to the task and should allow the user to complete the task without having to refer to information not included in the window.

Figure 8-16 presents an example of a window that places an unnecessary memory load on users. This window allows users to select the colors to be assigned to countries appearing on a map display. Users select a set of countries, choose a color, and then execute the selection on the map; the selections are cleared from the window, and users repeat this process for the next color. The window design and selection process require users to know the country abbreviations and to remember both the countries they have already selected and the colors they have assigned to these countries. In addition, users have to check the map display to ensure they do not inadvertently select a country a second time and assign an incorrect color to it. A better design for this window would be to present a matrix of countries and colors so that users can make all of their selections at once. This design would minimize the memory load placed on users and reduce the likelihood of user error because all of the information would be available in the window and users would be able to review their selections as they made them.

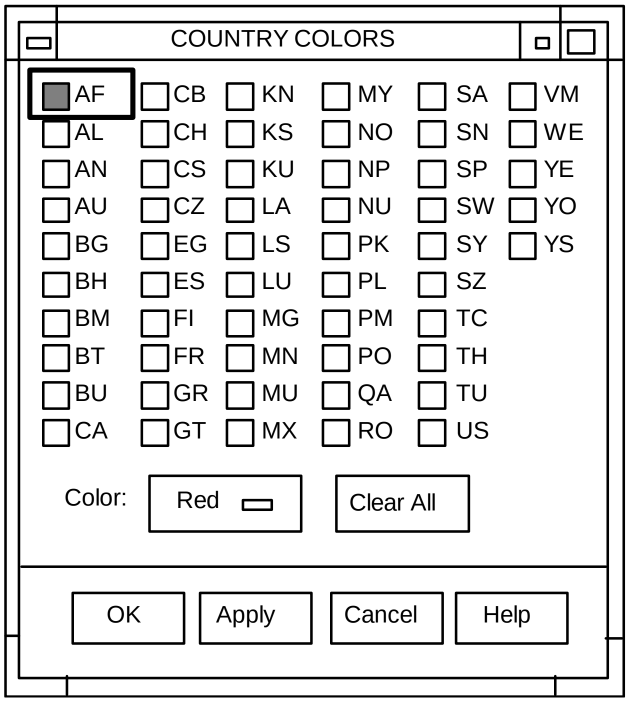


Figure 8-16. Example of inappropriate window design that places unnecessary memory load on users (from the JMCIS style guide).

Applications should be designed so that users enter all of the information pertaining to a particular operation in a single window; users should not be presented with a new window each time they enter a new data value and should not have to cross-reference one window with another or remember information from one window in order to complete another. When related information is spread among multiple windows, users are required to remember what they saw in early windows as they work in subsequent ones. Incorporating all of the information in a single window eliminates this memory requirement and reduces the opportunity for user error.

8.5.5 Techniques for Coding Information

This section addresses coding techniques that can be used, either singly or in combination, to convey meaning in applications. Developers should select the available coding dimension(s) most appropriate to the task being performed by users and then apply these consistently throughout the application.

8.5.5.1 Capitalization

Capitalization should not be used as the sole indication of critical information in a window. Attention should be focused on critical information by bolding/brightening, color coding, or other means. Both color (e.g., red) and sound (e.g., audio alarms) should be used for critical messages, and users should be required to respond before an audio alarm is terminated.

8.5.5.2 Color

Color should be used only as necessary to provide required functionality. Coding methods other than color should be applied whenever possible. The addition of color can increase response time and the likelihood of error due to color confusions. Section 8.3 provides guidelines on how to use color to highlight and redundantly code critical information in windows. Color is a more effective code for search tasks and symbol identification tasks than other cues such as shape, size, or brightness. The performance advantage of color coding increases with the density of the symbols in a display and when the number of nontarget symbols of a different color than the target increases. However, color used inappropriately in displays can degrade user performance when compared to monochromatic displays.

8.5.5.3 Flashing

Flash coding should be used only to display urgent information for user attention. No more than two levels of coding should be used. MIL-HDBK-761A indicates that the flash rate should be in the range of 3-5 Hz with equal on and off intervals; if two flash coding levels are used, the second should flash at 1-2 Hz. IEEE recommendations indicate the flash rate should be within 1-5 Hz with a minimum cycle of 50 percent. When flash coding is used on a displayed item, a flashing marker symbol (such as asterisks) should be used rather than blinking the information itself. Users should be able to acknowledge the event causing the flashing and suppress it if desired.

8.5.5.4 Reverse Video

Motif uses reverse video to indicate that an item has been selected. As a result, developers should limit their use of this attribute for coding to prevent confusion by users. Also, although effective in making data stand out, reverse video can reduce legibility and increase eye fatigue.

8.5.5.5 Size and Shape

If size coding is used in an application, the number of size codes should be limited to five or less, and users should be required to interpret relative size rather than absolute size. Alternatively, users' perception of object size can be manipulated by varying the color saturation and lightness of the object (see section 8.3.3 on color selection). If shape coding is used, the number of shape codes should be limited to 10-20, and the shapes used should relate to the object or operation being represented. Developers should apply a minimum number of colors within the shape and provide the minimum amount of detail necessary for users to identify the meaning assigned to the shape.

8.5.5.6 Sound

Applications shall use auditory signals to alert users to critical conditions or operations. If auditory signals are used for noncritical operations (e.g., as an alternate means of information presentation), users should be provided with a simple means to acknowledge and turn off the signal at their discretion. Auditory signals should be intermittent in nature and allow sufficient time to respond; they should be distinctive in intensity and pitch, and the number of signals provided to users should not exceed four. The intensity, duration, and source location of the signal should be selected to be compatible with the acoustical environment of the users and the requirements of other personnel in the area surrounding the system. In addition, users should be able to adjust the volume of the signal and to specify different sounds as feedback for different events as user-selectable settings.

8.5.5.7 Text Font and Styles

User attention can be drawn to text information through the use of different fonts and styles (e.g., boldface, italics, all upper-case letters). Developers should not use more than two styles of type (e.g. regular and italics) or two weights (e.g., regular and bold) at one time.[56] In addition, variations in type sizes should be minimized and limited to no more than three at any one time. Upper-case letters should be used for headings and labels and to emphasize words or phrases in continuous text; however, capitalization should not be used as the sole indication of critical information in a window. While underlining is also effective in drawing user attention to specific text information, it can reduce legibility and so should be used sparingly. In addition, in hypermedia-type software tools, the underlining of a word or phrase is used to indicate the

[56]. Because these specifications recommend the use of a heavy (i.e., bold) font for general readability, developers may be restricted in the number of text styles available for use in an application.

presence of a link to other information. If underlining is used in an application, it should be implemented in ways that do not conflict with this evolving convention.

8.5.6 Dynamic Information in Windows

When a window contains dynamically changing information, an application should allow users to control the rate of update based on the use to be made of the information. In addition, an application should allow users to freeze the display of any information that is being updated automatically and to resume the updating either at the point of stoppage or at the current point in time.

MIL-HDBK-761A recommends that when dynamically changing information must be reliably and accurately read, the information should not be updated more often than once per second. If users have to identify the rate of change or read gross values, the information should be updated at a rate of between two and five times per second.

Applications should be able to alert users to critical information that becomes available in an inactive or minimized window in the application. If a dynamic window is temporarily frozen (e.g., while executing a print command), the application should provide immediate feedback to users and should prompt them to return to automatic updating. Users should be informed if significant changes in data occurred while the display was frozen.

8.5.7 Consistency in Design Across Windows

Application developers should adopt a consistent organizational scheme for the key elements in a window and then apply that basic scheme to all windows in the application. The same window design should be employed whenever users have to perform the same basic task; for example, a single design should be used to present identifying information on data records, whether the data are tracks or messages. Different or distinctive elements can appear in a window to fit the task being performed, but these elements should be consistent across windows within the application.

8.5.8 Exiting an Application

Developers should identify the windows from which users are likely to exit the application and provide the ability to do so only within these windows. If the primary window(s) of an application include a menu bar, the ability to exit the application should be available as an option in the appropriate menu in the menu bar. Primary windows without a menu bar should provide an Exit push button if users need to be able to exit the application from the window. An Exit push button should be provided in windows only where it makes sense for users to take this action; including this capability in all windows (so that a window contains both Close and Exit push buttons) is not recommended because users may confuse the actions executed by these controls and inadvertently exit the application, resulting in a possible loss of data.

Developers should consider how access to an application will be provided within a system from the user’s perspective and then determine the most appropriate means to exit the application from that perspective. Applications that are available as standalone modules should provide an exit capability as described above so that users have to execute an explicit action with regard to quitting the application. However, when applications are closely coupled within a system, users may not be aware that they have changed applications as they work in different windows. As a result, the requirement to exit an application may be misleading given the context in which the application is used. In this situation, developers should consider mapping the Close (or Cancel) button to the actions normally performed by Exit.[57]

8.5.9 Designing to Minimize User Error

Applications should be designed to minimize the opportunity for user error. Users should be able to perform only legal operations, rather than allowed to execute an incorrect operation and then informed that they have made an error. Only those actions that are relevant should be available for user selection. Controls that cannot be selected should be both visually deemphasized by graying out to indicate their unavailability as well as disabled (so that no action is executed if they are selected by users). Applications should provide visual and behavioral cues to prevent users from performing illegal or incorrect operations, rather than relying on error messages to inform users after errors have been made.

Developers can minimize user error by ensuring that for each object displayed by the application, the actions available match the function users are expected to perform with the object. For example, if users are required to select a single item in a list box, the available selection methods should be limited to this type of action. Users should not be able to select multiple items in the list, then be informed in an error message window that they should have selected only one item.

Applications that define different processing modes (see section 3.5.8 on use of modes) increase the likelihood of errors because users have to remember how the application behaves in each mode, especially if there are no visual cues present to indicate the mode currently in effect. To minimize these kinds of errors, developers should not include controls that perform different functions depending on the current mode in their applications.

While the opportunity for user error should be minimized, users should not be unnecessarily constrained when working in an application. For example, a window containing several mandatory data fields might gray out the OK or Save push buttons until these fields are filled so that users are unable to save the data before completing the mandatory fields. This approach allows users to fill the fields in whatever order they choose or to make corrections as desired before attempting to save the data. A more constraining approach would manage each of

[57]. The specifications presented here differ from the DoD style guide which recommends the use of Exit when quitting an application. The DoD style guide assumes that applications are not as tightly integrated within a system as may occur in a GCCS-based system.[58]. The implementation of online documentation, job planner and “how to” modules, and computer-based training and computer-managed instruction services will be specified in other GCCS documentation.

the fields so that users have to enter a value before being allowed to leave the field. This approach forces users to perform data entry in a lock-step sequence defined by the application and restricts their ability to perform the task in a manner that does not match this sequence. The application should be designed to provide flexibility and support the user's sense of control, while at the same time bounding user interaction to reduce the opportunity for error.

8.5.10 Designing for Procedural Efficiency

Applications should be designed to minimize the number of actions users are required to execute in order to complete an operation. Efficiency can be improved by minimizing the number of operations that users have to perform in a window. For example, applications should prefill text entry fields (e.g., current date, ownship name, position) with default data values whenever possible. Similarly, applications where the same data fields are repeated in multiple windows should require users to fill the field once and then automatically place this value in the field as the default whenever it recurs.

Procedural efficiency can also be improved by matching the window design to the task being performed and minimizing the number of intermediate windows that users must interact with in order to complete the task. An application should be designed so that users are able to perform each major task in a single window containing all of the information relevant to the task. Developers should be sensitive to problems created by presenting users with a sequence of independent windows, each of which contains only a portion of the overall task being performed. While this type of design may be efficient from a software development perspective (e.g., by allowing re-use of previously designed windows), it can produce an unnecessarily complex task navigation problem for users, increase the difficulty of the task being performed, and place an unnecessary memory load (see section 8.5.4) on them to remember information not included in the current window.

Developers should consider the manner in which an application will be launched once it is integrated within a system. As indicated in section 7.3.1, the functionality provided by an application may be available as a single menu option or as a a group of options distributed among multiple menus. Applications, especially those that are chart-related, should be designed so that a task-related window is displayed when users launch the application; applications should not create an intermediate window whose sole purpose is to provide users with access to application functions.

8.6 WINDOW MANAGEMENT CONSIDERATIONS

8.6.1 Initial Window Appearance

Application developers should determine the appropriate appearance for each application window when it first appears on the screen. Developers should consider what information should be visible. For example, when a scrollable window appears on the screen, the window should display the topmost part of the information and should be wide enough for users to read a complete line of text without scrolling.

In addition, developers should determine the appropriate settings for the controls in a window when it appears on the screen. For example, when radio or check buttons are included in a window, developers should determine if the application should save the button settings in their most recently executed state or return them to a default (or unselected) state when the window next appears on the screen. However, when users select button(s) but do not execute the selection, the application should not save the selection and the button(s) should revert to their original state when the window was displayed.

8.6.2 Initial Window Size and Placement

When a window is displayed, it may fill all or part of the available screen in the system window. An application window that covers all of the screen should contain frame borders so that users can resize it and access any other windows that it might cover.

Whenever possible, each window in an application should be sized so that all of the objects in the window are visible when the window first appears and users can see the whole page with which they are working. At a minimum, a window should be wide enough for users to read the title, and should be tall enough for users to read the information in the title bar and menu bar.

Application developers should determine the initial placement for each window when it first appears on the screen. Thereafter, a window should be displayed at the same location where it was when it was last closed or minimized. In deciding on a window's placement, developers should consider the importance of the information presented in the window (e.g., important information should be placed at the center of users' visual focus), the other information already displayed (e.g., important information should not be obscured), the distance from the current pointer location (e.g., the amount of pointer movement required by users to execute an operation should be minimized), and, in the case of dialog windows, the information to which the window relates (e.g., a dialog window should be positioned to the right of the information to which it relates).

When a window is initially displayed, it should be positioned on the screen so that it is completely visible. If the window is being displayed on top of another window, the new window should be offset below and to the right of the existing window so that the title of the window underneath remains visible. If a new window (e.g., a dialog window) is related to other windows already displayed, it should be positioned so that it does not obscure information (e.g., title, text, or controls) in the other windows. As indicated above, whenever possible, the window should be positioned to the right of the information to which it relates. If space is insufficient, the window should be displayed to the left, below, or above the information. If a window is not related to other windows currently open, it should be centered on the screen when displayed.

8.6.3 Resizing and Maximizing

When users change the size a window, only the location of the window borders should change, not the size of the objects within the borders. For example, if a text window is enlarged, more lines of text should be visible in the window, but the size of the text should not change. In addition, when a window is resized, the relative position of the objects in the window should not change. When a window is made smaller, it should either clip the area that can no longer be displayed or provide scroll bars for viewing this area. Developers should allow the size and/or position of objects within a window to change only if the functionality of the application (e.g., imagery manipulation) requires this capability to be available.

The contents of the window should remain visible during resizing so that users can view the effect of the change in size on the amount of information that can be seen. Horizontal and vertical scroll bars should appear as appropriate when a window is resized to be smaller than the information being presented in the window. In windows that have multiple panes, application developers should determine how the space in the window is divided between panes after users resize the window. Depending on the functionality (e.g., mapping), an application may require the size and/or contents of the panes to change automatically based on the size of the window.

The extent to which a window can be resized smaller should be limited so that objects are not obscured from view when the window is at its minimum size. In addition, the extent to which a window can be resized larger should be limited so that the classification and system menu bars cannot be obscured. Likewise, the maximum size of a window should be defined so that these portions of the system window are not covered when an application window is maximized.

8.6.4 Order of Window Interaction

Application developers should determine for each dialog window in the application the set of other windows with which users are allowed to interact while the dialog window is displayed. Developers should specify if interaction with other windows is unrestricted (i.e., modeless) or if interaction is not allowed with any parent window (i.e., primary modal), any window in the application (i.e., application modal), or any window in the system (i.e., system modal). In addition, developers should determine whether users should be able to access some of the objects within a parent window while working in a child window or whether the parent should be closed when the child is displayed.

8.6.5 Processing in Minimized Windows

Applications should be designed so that users are kept informed when processing is ongoing in a minimized window. Users should be informed whenever critical processing events (e.g., when processing stops, when an error occurs, when additional input is needed) occur in a minimized window.

9.0 TASK-SPECIFIC WINDOWS

9.1 HELP WINDOWS

9.1.1 Access to User Support Resources

On-line support resources shall be available to users at both the system level and within individual applications. These resources include access to help, online documentation, job planner and "how to" modules, and computer-based training and computer-managed instruction services. This style guide provides specifications for the delivery of help by systems and applications, with the implementation of the remaining support resources defined in other system documentation.[58]

Each system shall include a Help menu in the system-level menu bar that provides access to system-level help options and to other support resources. Help-related options to be provided by each system include help on the system, help on the keyboard, and system navigation aids (see section 7.3.6). Help on the system should present information on system capabilities, including a list of software modules (with version numbers) available on the system. Help on the keyboard should provide information on function keys, mnemonics, and keyboard accelerators for the platform(s) on which the system is installed. Application developers shall provide help support for each application window as defined in section 9.1.4.

The Motif Style Guide defines two models for the contents of a Help menu and recommends that application developers implement one of the models, to include only those help functions actually supported by the application. In systems complying with the specifications presented here, the primary window(s) in an application may not include a menu bar from which to provide access to these functions. Furthermore, the system integration process may result in an application being decomposed into components that are distributed among multiple menus based on individual functions performed. As a result, the specifications presented here focus on delivering application-level help support via help windows, and recommend that applications implement one of the two Motif help models.

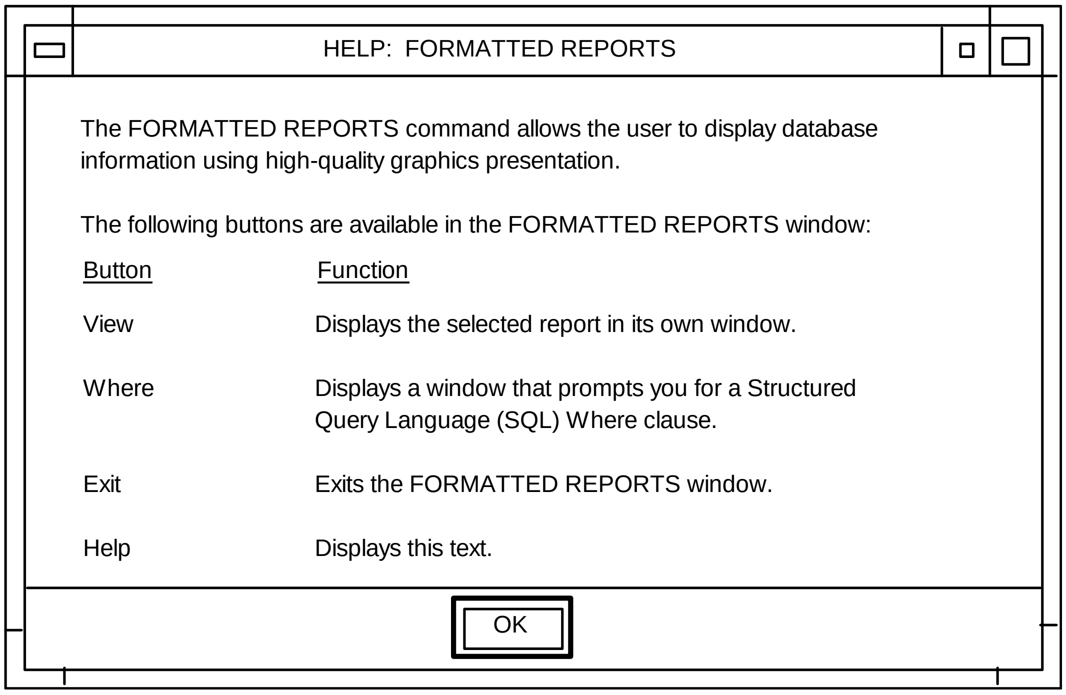
If an application provides any of the functions defined by the Motif help models, it shall be implemented in accordance with the Motif Style Guide. In particular, if context-sensitive help is supported, it shall be available when users place the pointer on a window element (e.g., a push button) and press <Help>, <Shift><Help>, or <Shift><F1>.[59] As an alternative to context-sensitive help, developers are encouraged to design application windows to include a message area that could be used to display additional information about the control that currently has input focus. For example, when the location cursor is displayed on a text field, the message area might provide range and format information to help users perform data entry correctly.

9.1.2 Window Design

A help window, such as that shown in figure 9-1, should include a window title that identifies the contents, a main area that is scrollable if the amount of information exceeds the size of the window, and an OK push button at the bottom of the window to remove it from the

[59]. <Shift><F1>, rather than <F1>, should be used as an alternate key mapping on keyboards that do not provide a <Help> key.

screen. The push button should appear and behave as the default action so that users can close a help window by pressing <Return>.


HELP: FORMATTED REPORTS

The FORMATTED REPORTS command allows the user to display database information using high-quality graphics presentation.

The following buttons are available in the FORMATTED REPORTS window:

Button	Function
View	Displays the selected report in its own window.
Where	Displays a window that prompts you for a Structured Query Language (SQL) Where clause.
Exit	Exits the FORMATTED REPORTS window.
Help	Displays this text.

Figure 9-1. Example help window (from the JMCIS style guide).

When a help window appears on the screen, it should be wide enough for users to read the text in the window; it should display the information at the beginning of the Help description and be placed on the screen so that the window does not obscure the object(s) it is describing. A help window should be displayed (in descending order of preference) to the right, left, above, or below the content for which help was requested. Help should not disrupt any ongoing processing in the window or application about which help was requested.

Users should be able to move and resize a help window. In addition, users should be able to keep the help window on the screen while working in an application window (e.g., so they can follow the sequence of steps listed in help). Users should be able to print the content of any help window, either by selecting all of the text in the window or by marking the beginning and end of the portion of interest. When done with help, users should have an easy means to return to the window(s) in which they were working when they requested help.

Developers shall provide help support for every primary, secondary, and dialog window in the application. This support shall normally be provided in a help window that users access by selecting the Help push button in an application window, or from the Help pull-down menu if the window contains a menu bar. If a help window is not available, the Help push button should be dimmed and unavailable for selection; users should not select Help and then be presented with a message that "No help is available for this window." If desired, applications using action icons can provide access to help via an action icon rather than a menu title. If an application window contains a message area, help support may be provided there, rather than in a separate help window.

9.1.3 Window Content

Each help window shall be titled HELP, followed by the name of the window for which help was requested; for example, if help is requested on the Formatted Reports window, the title of the help window should be "HELP: FORMATTED REPORTS." A help window should be able to be moved and resized, and should be modeless.

A help window shall contain a read-only text display area, with an OK push button centered at the bottom of the window (see figure 9-1). Whenever possible, the window should be large enough for all of the information to be visible in the text display area. At a minimum, the window size should allow at least ten lines of text to be visible in the display area and should be wide enough to display an entire line of text. A help window shall contain a vertical scroll bar if the length of the information being presented exceeds the size of the text display area. A help window should not include a horizontal scroll bar unless system requirements dictate that these windows be reduced in size. A help window may also include:[60]

- Next and Previous push buttons (in addition to OK) if the help information included in the window is lengthy and should be viewed in page rather than line increments.

- An index (in the form of a list) at the top of the window (with the text display area below) to support rapid location of specific information about an application window that is extremely complex in content (e.g., contains multiple columns of information about which users are likely to request information) or functionality.

The nature of the information presented in a help window shall match the type of application window (i.e., primary, secondary, or dialog) for which it provides support. A help window shall present information relating to the application window for which it provides support; information on how to perform operational tasks (e.g., generate a report, execute a what-if analysis) shall be available in job planner and "how to" modules accessed from the system-level Help menu.[61]

Four areas of help content are defined: Purpose, Available Actions, Procedures, and Supplementary Information. The information to be presented in each content area is described below. Table 9-1 indicates whether a content area is required, recommended, or

[60]. A help window should not include push buttons linking to other support modules if doing so will launch processes for these modules that will negatively impact system performance.

[61]. Design guidelines for these modules will be provided in other GCCS documents.

optional (i.e., as needed to support task performance by users) for the types of windows in an application. If a help window contains information in these content areas, it shall be presented in the order indicated below and titled with the appropriate section heading.

Table 9-1. Help content for primary, secondary, and dialog windows.

Content Area	Primary Window	Secondary Window	Dialog Window
Purpose	Required	Required	Optional
Actions Available	Required	Required	Required
Procedures	Recommended	Optional	Optional
Supplementary Information	Optional	Optional	Optional

- The Purpose section provides a short description of the task(s) that users perform in the window. Each task is described both as it relates to the functionality provided by the application and from an operational perspective. While the purpose of a window can usually be described in a single sentence, longer statements may be needed if a window allows users to perform multiple tasks. The purpose section also identifies if the window is targeted to a particular category of system or operational user (e.g., a database administrator or FOTC coordinator).

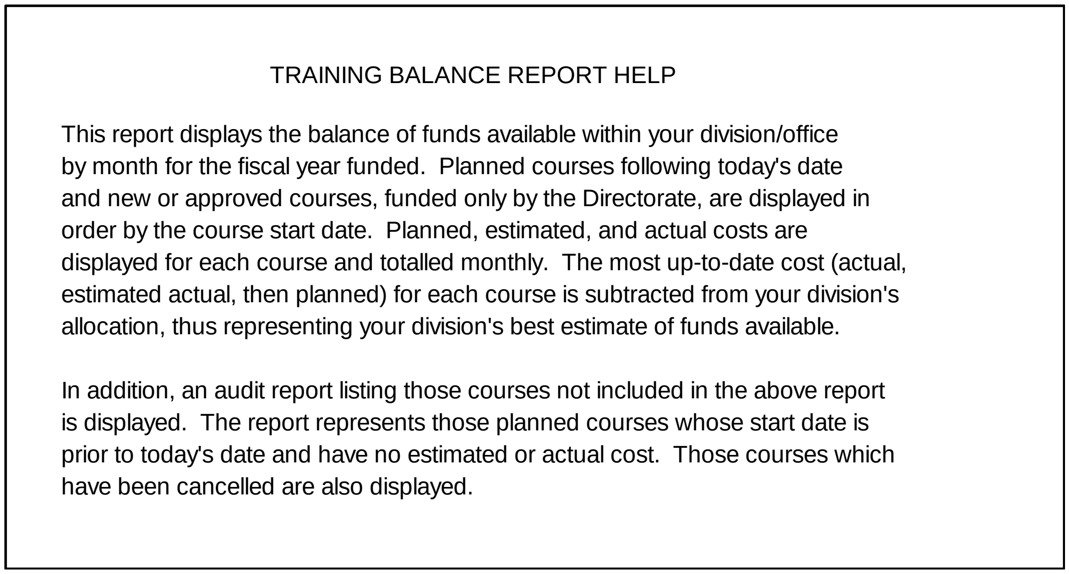
- The Available Actions section explains each of the actions that can be executed in the window (excluding those pertaining to window management functions). In most cases, the actions are those available in the push buttons in the window but also include the action options in any pull-down and pop-up menus if these are components of the window. The description of each action focuses on its impact on the task being performed in the window; however, if appropriate, the description should also explain if the action affects other portions of the current application, other applications, or the system as a whole.

- The Procedures section provides a step-by-step explanation of how users execute the task(s) presented in the window. Each step describes both the user action, such as pressing specific keys or selecting specific controls, and the system response, such as changes in screen appearance (if any) resulting from the action. This section also describes each step within an operational contextespecially if the task(s) performed in the window have consequences in terms of a larger operational function. Where appropriate, this section identifies any other windows that should be accessed in order to perform other related aspects of the function currently described (e.g. what other information or actions does the user need to be aware of/execute in order to complete the function in the current window).

- Depending on the task(s) performed in the window, the Supplementary Information section can provide (1) the data source (e.g., database) from which information retrieved by the system and displayed in the window was drawn, (2) a glossary that decodes any acronyms used in the window, (3) explanations on how to recover from errors, and (4) a description of any shortcuts for procedures normally executed in the window.

The text in a help window shall be presented in mixed case (i.e., an appropriate combination of upper- and lower-case), with capitalization of words and phrases used to emphasize or highlight significant information. The text shall be single-spaced, with a double space between paragraphs. Sufficient space (e.g., at least one blank line) or a separator (e.g., a series of dashes) shall be used to visually separate the text in each content area. Subheadings may be included as needed to help structure and organize the information being presented. Whenever possible, text should be displayed in bulleted form, steps (e.g., in a procedure) should be numbered sequentially, and explanatory information should be presented in columnar form (as in figure 9-1) so that users can find what they are looking for quickly and easily. Graphics should be used to supplement the text in a help window only if essential to understanding the task to be performed in the application window.

The information should be worded concisely, using consistent grammatical structure, phrasing, and punctuation. For example, figure 9-2 provides an example of inconcise wording; figure 9-3 presents a more concise version of the same information. When presenting a sequence of steps, the explanation should follow the same sequence (e.g., save data before quitting; or save data, then quit the application).

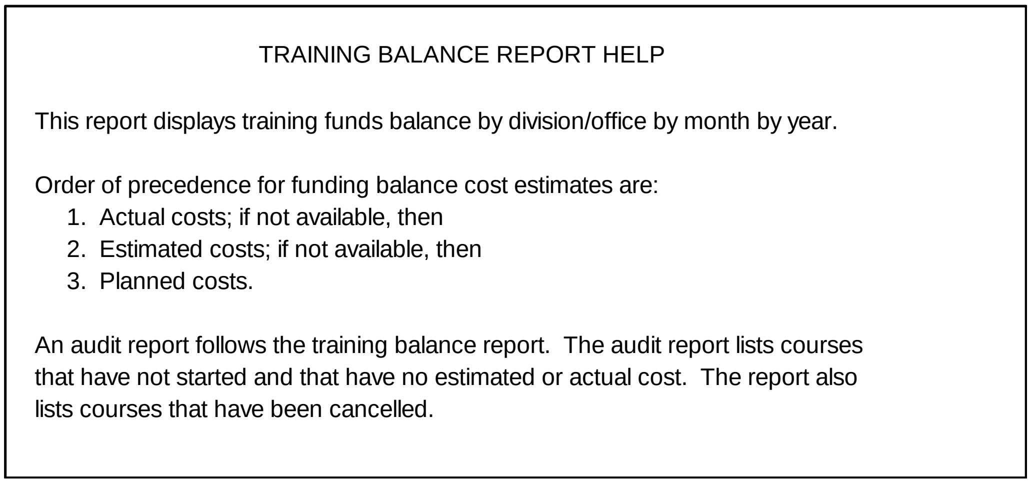


TRAINING BALANCE REPORT HELP

This report displays the balance of funds available within your division/office by month for the fiscal year funded. Planned courses following today's date and new or approved courses, funded only by the Directorate, are displayed in order by the course start date. Planned, estimated, and actual costs are displayed for each course and totalled monthly. The most up-to-date cost (actual, estimated actual, then planned) for each course is subtracted from your division's allocation, thus representing your division's best estimate of funds available.

In addition, an audit report listing those courses not included in the above report is displayed. The report represents those planned courses whose start date is prior to today's date and have no estimated or actual cost. Those courses which have been cancelled are also displayed.

Figure 9-2. Example of inconcise wording of help information (from MIL-HDBK-761A).



TRAINING BALANCE REPORT HELP

This report displays training funds balance by division/office by month by year.

Order of precedence for funding balance cost estimates are:
1. Actual costs; if not available, then
2. Estimated costs; if not available, then
3. Planned costs.

An audit report follows the training balance report. The audit report lists courses that have not started and that have no estimated or actual cost. The report also lists courses that have been cancelled.

Figure 9-3. Example of concise wording of help information (from MIL-HDBK-761A).

The information presented in a help window should assume that users have received previous training and need to be reminded of the proper information or procedure. Users should be able to easily scan to the text of interest (e.g., capitalize it to make it easier to find), and the steps should be numbered and presented as separate paragraphs if more than two steps are involved. Help should decode all abbreviations and acronyms used in the window about which users requested help, and should explain how each user-selectable object in the window behaves. A help window should be removed from the screen when the window about which users requested help is minimized or closed.

9.2 DATA ENTRY WINDOWS

9.2.1 Window Design

A data entry window, such as that shown in figure 9-4, consists of a template that is used to display, enter, change, and delete data. The template provides a set of data fields, each consisting of a field name and an area where the data value appears. The data fields may be text fields, option menus, or information-only areas, depending on the type of data being presented.

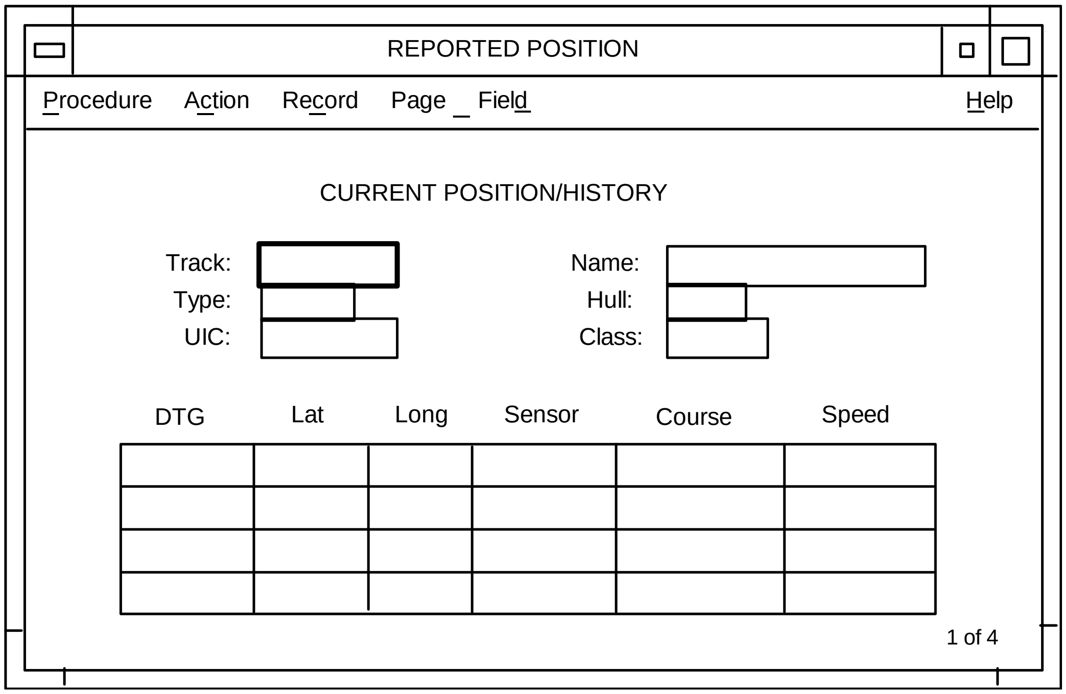
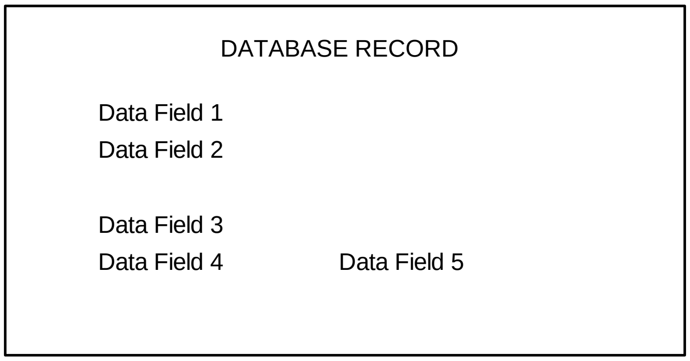


Figure 9-4. Example data entry window (from the JMCIS style guide).

A data entry window should include a title that describes the contents of the window. If the data fields extend beyond a single page (i.e., what can be displayed in the window without scrolling), each page should have the same title and be numbered to indicate the current page and total number of pages (e.g., 1 of 3). If possible, related data fields should appear on the same page. The numbering of items should be continuous from one page to the next (e.g., if the last numbered item on page 1 is #5, the first numbered item on page 2 should be #6).

The design of each data entry window should be consistent with the task being performed by users. Data fields should be organized by sequence of use, frequency of use, or importance, with related fields appearing together and separated from unrelated fields. If users are entering data from a hardcopy form, the window format should be identical to the hardcopy format, as shown in figure 9-5. When a data entry window contains different kinds of data fields (e.g., text fields, option menus), the data fields should be arranged to minimize the amount of hand movement between pointing device and keyboard required of users who choose to use both input devices to perform data entry.

Hardcopy Format:



Window Format:

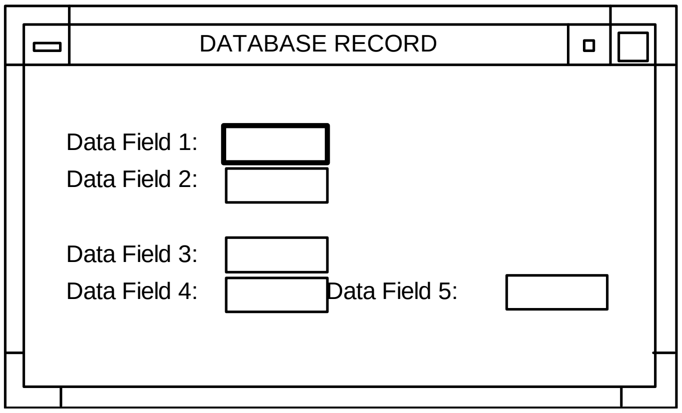


Figure 9-5. Consistency in layout of hardcopy form and data entry window (from the DoD style guide).

Windows that require users to enter data in tabular form should contain data fields arranged in rows and columns, with each row and column labeled; users should not be presented with an empty text window, with no clues as to format, for entering the information. If the data fields extend beyond a single page, the row and column labels should remain along the edges of the window and should not disappear from view when the window is scrolled. Every fifth row of data should be separated by a blank line or other delimiter. The information in each field should be automatically justified, and numeric data should be automatically justified according to the decimal point.

A data entry window should contain the controls (such as push buttons) or pull-down menus needed to support data entry and manipulation using either the pointing device or keyboard. For example, if a data entry window takes more than one page, the window should include controls that provide for paging. In addition, if users have to make multiple entries, the data entry window should include controls to clear the window and restart data entry. If an application uses push buttons as the controls in a data entry window, they should appear along the bottom of the window and be visually separated from data entry areas.

A data entry window should be designed so that users can obtain information about a data field and its contents. Options available for providing this information are to automatically display it in the message bar of the window when the text cursor is placed in a field or to have users click on the data field name, select a Help push button in the window, or select Help from an application-level menu or action icon.

9.2.2 Data Field Format

Each data field name used in a data entry window should be unique, and the same name should be applied to the same field throughout the application. Data fields should be identified as mandatory or optional, and optional fields should be labeled as such. Data fields with values that can be changed should appear as a text field or option menu, depending on whether users have to type the value or can select it from a limited set of values. Data fields with values that cannot be changed should present the values in an information-only area.[62] Data fields for date/time and latitude/longitude should follow the formats indicated in section 8.4.6.

Groups of related text fields in a data entry window should be justified in one of two ways: Left justify the labels and left justify the text fields, or right justify the labels and left justify the text fields (as in figure 9-4). In both cases, the colon at the end of the label should be separated from the text field by one space. A conditional (or dependent) field should be placed to the right of or below the field to which it relates. The field may be either unavailable (i.e., grayed out) or not displayed at all until the control to which it relates is selected.

9.2.3 Data Entry and Manipulation

Users should execute the same set of actions to enter and manipulate data in data entry windows throughout the application. When a data entry window is initially displayed, the text cursor should be positioned at the beginning of the first data entry field in the window. Users should be able to place and move the text cursor using either the keyboard or the pointing device. Users should perform text entry in text fields as indicated in sections 2.3.3 and 6.4.2.1, move between text fields as indicated in section 6.4.2.2, and make selections from option menus as indicated in section 5.4.2.

Users should be able to tab to adjacent fields, and tabbing should move through the fields in the same direction (e.g., left to right, top to bottom) in all data entry windows in the application. Users should be able to back up to any field and change its contents prior to entering the data into the system. Tabbing should skip over the labels for data fields. Users should be able to execute an Undo command (e.g., as a keyboard accelerator) in order to revert the text in the

[62]. The DoD style guide recommends that updatable (i.e., editable) fields should be distinguished by underscores below the data field and that highlighting and colors may also be used. The specifications presented here use color to distinguish editable from noneditable text.

field currently being edited to its original content prior to the start of text entry. If a data field contains a default value, users should be able to accept the value by tabbing to the next field in the window; tabbing should not affect (e.g., remove) a default value in the field. If users modify the default in a field, the change should not affect the value the next time that it appears in the field.

9.2.4 Data Validation and Error Checking

The application should perform a validity check on the data entered and provide a visual and/or auditory error message when an invalid entry is detected. Syntactic error checking should be performed to ensure the data are in the correct format; for example, numeric data should meet range (minimum-maximum) requirements, and abbreviations and acronyms are in the proper format. Semantic error checking should be performed to ensure that the meaning or context of information entered is correct. This form of error checking compares the value in one data field against the value in another to ensure consistency and correctness.

An application should perform syntactic error checking at the time the user tabs out of the data field. If an error is detected, the field should be marked (e.g., by displaying the text in red, displaying a message in the message area of the window) but users should not be prevented from leaving the field. Instead, they should be allowed to review and correct the error(s) when they choose to do so. An application should perform semantic error checking as soon as possible and provide similar notification of errors. Because this form of error checking compares the entries in different fields, it is possible for a value detected as an error early in data entry to be corrected by subsequent data entry. As a result, users should be provided with a summary of errors (e.g., in a message window) at the time they attempt to enter the data into the system.

When users finish making entries in the appropriate data fields in the window, they should enter the data into the system with an explicit action such as selecting a Save menu option or an Apply or OK push button. Users should be able to enter what they have typed into the system at any time. When users accept the data entered in the window, all of the data in the window should be saved, regardless of where the text cursor is currently positioned within the window. If appropriate, the application should provide feedback to users (e.g., a confirmation message) to indicate successful data entry.

9.2.5 Data Query

If an application provides a data query capability, the language provided to users should reflect the structure of the data as perceived by users. The language should allow users to specify the data to retrieve from the database, then to display and manipulate (e.g., edit, delete) it as appropriate. Users should be able to modify portions of a query easily and without affecting other parts of the query. Users should be able to indicate what data are requested without having to tell the application how to find it. In addition, they should be able to construct queries using terminology that is operationally meaningful rather than using terms or formats that reflect how the data are stored in the database. The query language should be able to recognize spelling variants, acronyms and abbreviations, and modifications in the format of a search term. The order in which users list search terms should reflect their importance, and the information retrieved in response to the query should be presented in this order (see section 9.4 on tabular data windows).

Users should be able to construct simple and complex queries that retrieve the desired information, create predefined queries, and save, retrieve, and execute these queries. Table 6-1 contains recommended vocabulary for constructing and executing queries. The language should permit users to construct alternate forms of the same query that correspond to the variations available in natural language. The application should prompt users to confirm the processing of a query if the time required for data retrieval will be excessive.

9.3 TEXT WINDOWS

9.3.1 Window Design

When a text document is displayed in a window (e.g., in a word processing application), the window should be wide enough to display an entire line of text without having to scroll the window horizontally. The window should provide scroll bars for scrolling vertically through the document, and a standard location (e.g., the message bar of the window) where users can determine their current location within the document (e.g., current page number). Multi-line text displayed in a text window should contain no more than 40-60 characters and be left-justified; text should not be right-justified because this spacing is harder for users to read.

Users should be able to save, access, retrieve, and rename text documents as well as be able to print the document. The user should be able to specify print options (e.g., font), the portions of the document to be printed (e.g., pages 2-9 of a 10-page document), and the printer where the document will be sent (if more than one printer is available). The system should display an acknowledgment of the print command and provide users with the ability to check the status of the printer (online/offline) and the printer queue.

9.3.2 Text Manipulation

It is expected that, where appropriate, systems will provide the full range of text manipulation capabilities (e.g., in a word processor) through the use of COTS software. When an application supports text manipulation, it should provide the basic capabilities described in this section.

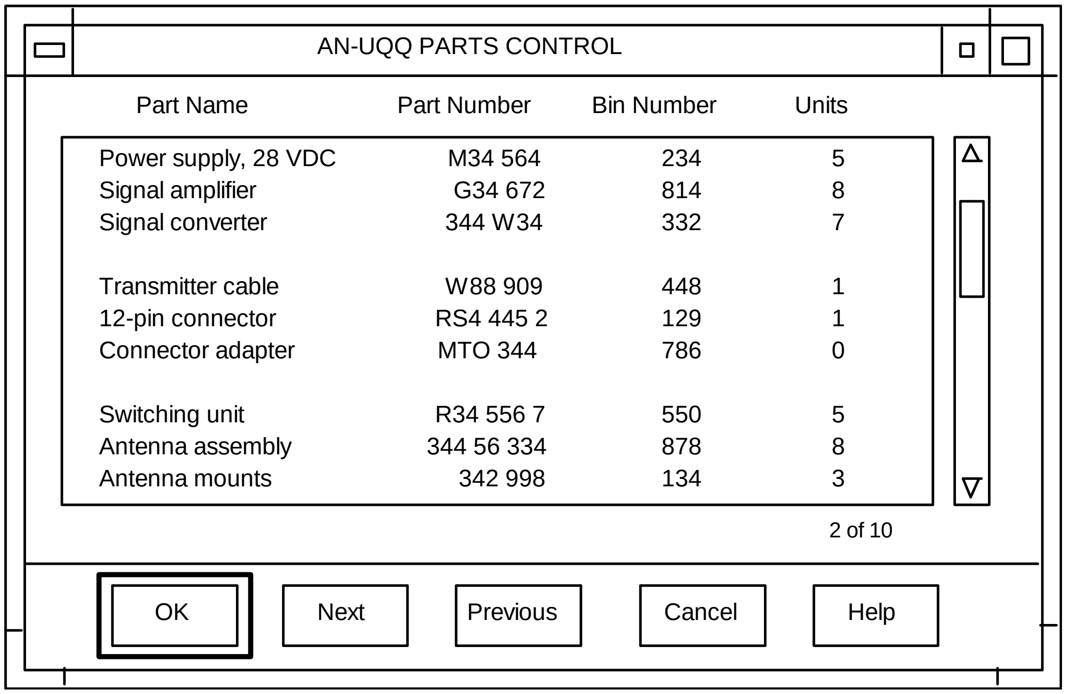
Users should be able to easily specify the format of a document (e.g., margins, tabs) and to select the font type, size, and style (e.g., bold, italics) for text. Automatic line break and wrap-around when the text being entered reaches the right margin as well as automatic pagination, with page numbers based on the number entered by users, should be available. The text that is displayed in the window should either be formatted to reflect how the document will appear when it is printed (i.e., what you see is what you get), or users should have the option to display the text in this format before deciding to print the document. A copy of the original document should be retained until users confirm that it is to be changed; the document should not be modified automatically as users make each editing change.

Applications should provide both search and search/replace capabilities for users. In the former case, users type the text string to be searched, and the application locates and highlights the first instance of the string in the text. The search should not be case sensitive. In the latter case, users type both the text string to be searched and the text string to which it is to be changed. The search should be case sensitive; e.g., users should be able to change "navy" to "Navy."

9.4 TABULAR DATA WINDOWS

9.4.1 Window Design

A tabular data window, such as that shown in figure 9-6, should be used to organize and display alphanumeric information in tabular or columnar form. This type of window should be used for information display only; a single- or multi-columned list box (see Section 6.5) should be provided if users are expected to select one or more of the items.



Part Name	Part Number	Bin Number	Units
Power supply, 28 VDC	M34 564	234	5
Signal amplifier	G34 672	814	8
Signal converter	344 W34	332	7
Transmitter cable	W88 909	448	1
12-pin connector	RS4 445 2	129	1
Connector adapter	MTO 344	786	0
Switching unit	R34 556 7	550	5
Antenna assembly	344 56 334	878	8
Antenna mounts	342 998	134	3

Figure 9-6. Example tabular display window (from MIL-HDBK-761A).

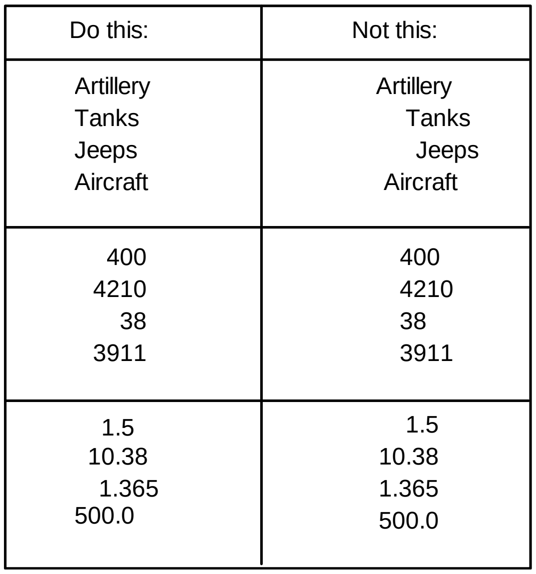
The window should include a title describing its contents, and vertical and horizontal scroll bars if the information being presented exceeds the space available in the window. If the data in a window extend vertically across several pages, each page should be numbered, and the window should include push buttons for paging commands. In addition, the columns should be labeled identically on each page, and the last line of one page should be the first line on the succeeding page. Whenever possible, the content of the window should be arranged so that it does not extend over more than one page horizontally. If the data being displayed are the output to a database query, the window should include the query that was executed.

If the information in a tabular display window can be scrolled horizontally, the window should be designed so that the column heading can scroll along with its associated column. If the information can be scrolled vertically, the column heading should be placed outside the scrollable area so that the heading remains visible when the column data are scrolled.

9.4.2 Window Content

Each column in a tabular data window should have a heading, and the data in one column should be clearly separated from that in other columns (usually by at least three spaces). Data groupings should be indicated with blank space, separator lines, and/or different intensity levels. Multiple colors should be used only if the colors provide additional meaning. If the window contains many rows and columns, a blank space should be inserted after every third to fifth row and column to facilitate scanning by users.

Alphabetic data should be left-justified within a column, numeric data without decimals shall be right-justified, and numeric data with decimals shall be justified by the decimal point, as shown in figure 9-7. Long strings of numbers should be delimited with spaces, commas, or slashes to facilitate readability, and leading zeros should not be used unless required for clarity. If the data in a record extend beyond a single line, each line beyond the first one should be indented or in some way identified as a continuation of the record.



Do this:	Not this:
Artillery Tanks Jeeps Aircraft	Artillery Tanks Jeeps Aircraft
400 4210 38 3911	400 4210 38 3911
1.5 10.38 1.365 500.0	1.5 10.38 1.365 500.0

Figure 9-7. Example of data justification within columns (from the DoD style guide).

The data in a tabular display window should be grouped or arranged so that users can identify similarities, differences, trends, or relationships. For example, depending on the purpose of the window, data may be presented in sequential, spatial, alphabetical, functional, or chronological order. Data that are particularly important, require immediate user response, and/or are used more frequently should be grouped at the top of the window. If the data extend across several pages, related data should be placed on the same page. If an application allows users to display the same information as they entered (e.g., in a data entry window), the format for the display should be compatible with the format used for data entry. The ordering and layout of the corresponding fields should be consistent, and the names of individual fields should be the same.

9.5 GRAPHIC DISPLAY WINDOWS

9.5.1 Window Design

A graphic display window should be used when presenting information in the form of a line graph, bar chart, histogram, or flowchart. The window should include a title describing its contents and should be sized whenever possible so that the entire graphic display is visible when the window appears on the screen; users should not have to scroll or resize a graphic display window to view its contents.

9.5.1.1 Line Graphs

Applications should use line graphs, such as those shown in figure 9-8, for presenting trend information, spatially structured information, time critical information, or relatively imprecise information. The axes of the graph should be clearly labeled and include the unit of measurement as appropriate. The labels should be in upper- and lower-case letters and oriented left to right (including the vertical axis of a graph) for normal reading. The minimum and maximum value should be indicated on each axis, with up to nine intermediate markings showing gradations on the axis. The starting point on each axis should be zero, with the gradations indicated in whole numbers, unless a zero starting point is inappropriate for the data being displayed. The gradations should be at standard intervals (e.g., 1, 2, 5, 10), with intervening gradations consistent with the labeled scale interval.

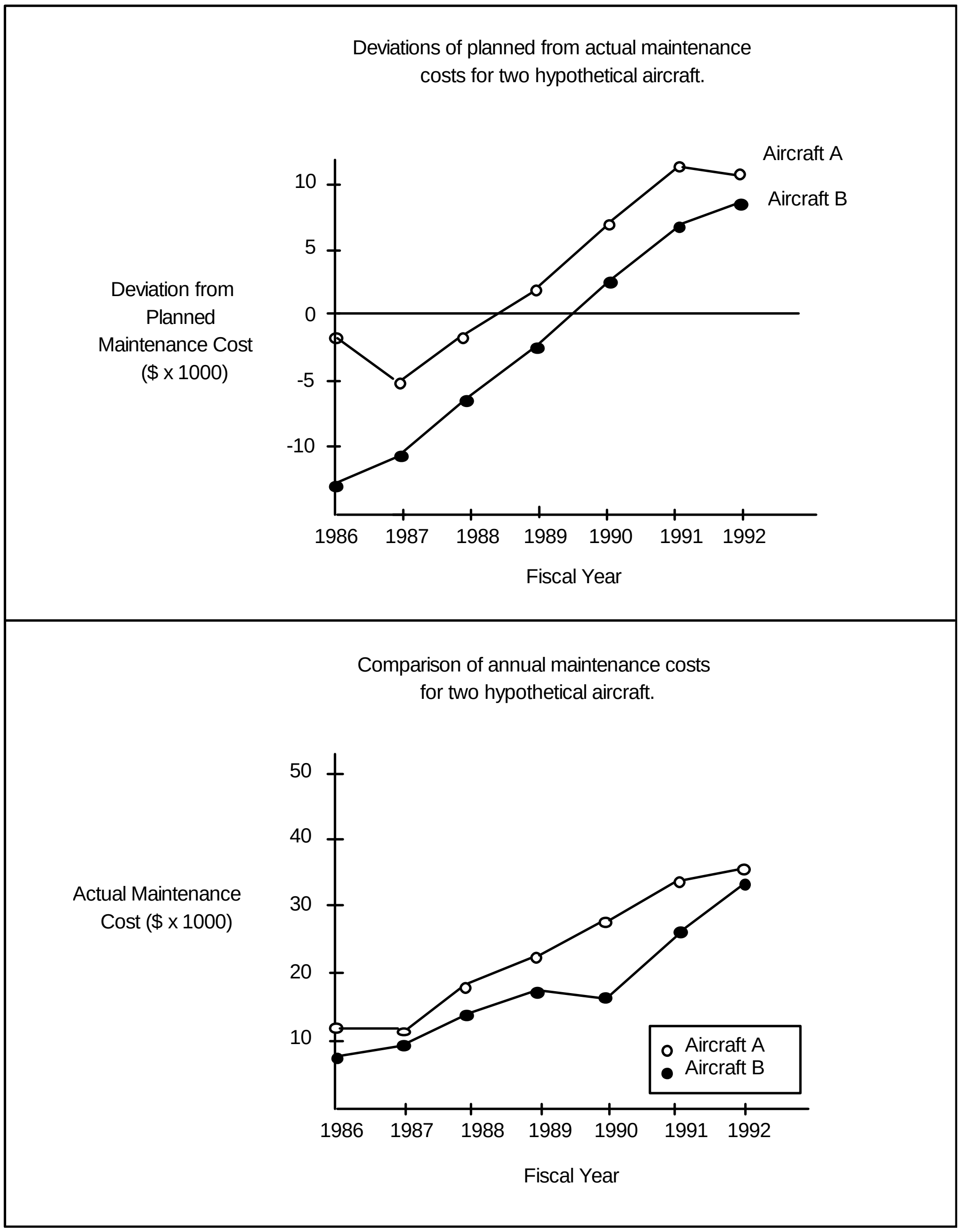


Figure 9-8. Example line graphs (from MIL-HDBK-761A).

Labels should be used instead of legends or keys when it is necessary to identify the data that are plotted on the graph. The labels should be oriented horizontally and located next to the data being referenced. Each line or curve on a graph should be labeled and coded (e.g., with solid, dashed, dotted lines), and a line or curve containing critical or abnormal data should be coded (e.g., by color, line thickness, annotation) to call attention to that part of the display. If grid lines are included in a graph, they should be unobtrusive and not obscure the data presented in the graph; users should be able to display or suppress grid lines as desired. The same coding scheme should be used consistently within an application whenever the same types of data are presented graphically.

A line graph should be limited no more than four or five lines/curves, with each one identified by an adjacent label (rather than in a separate legend). If corresponding data are presented in multiple graphs, the same coding scheme should be used in each graph. Coding should also be used to highlight more important or critical information or to identify actual from projected data within the graph.

If users are required to compare multiple trend lines, the lines should be presented on a single graph. If the lines have to be presented in multiple graphs, users should be able to redraw the graphs using the same scale on both graphs to facilitate comparison. If users have to read precise values from a graph, users should have the option to display the actual data values on the graph and to zoom the display if necessary. In addition, the application should provide aids for scale interpretation (e.g., displaying a grid upon request, providing vertical and horizontal rules that the user can move to the intersection point, or letting users click on a point on the graph and having the exact values displayed in a pop-up window).

A surface chart is a type of line graph in which the data being depicted represent all parts of a whole. The curves/lines are stacked above one another to indicate aggregated amounts, and the area between each curve/line is coded using different colors, shadings, or textures and identified by a text label displayed within the area. If a surface chart is used, the data categories should be ordered to reflect the logical organization of the entity being displayed. If no a priori organization exists, the data categories should be ordered so that the least variable ones are at the bottom and the most variable at the top.

9.5.1.2 Bar Charts and Histograms

Applications should use a bar chart, such as the ones shown in figure 9-9, for comparing a single measure at several intervals, and a histogram for comparing a single measure when the number of intervals is large. The bars in a related set of bar graphs should have a consistent orientation (vertical or horizontal), and bars containing data that must be compared should be presented adjacent to one another. Frequency counts are usually displayed in vertical bars, and time durations in horizontal bars. If the displayed data have to be compared with a critical value, a reference index should be provided.

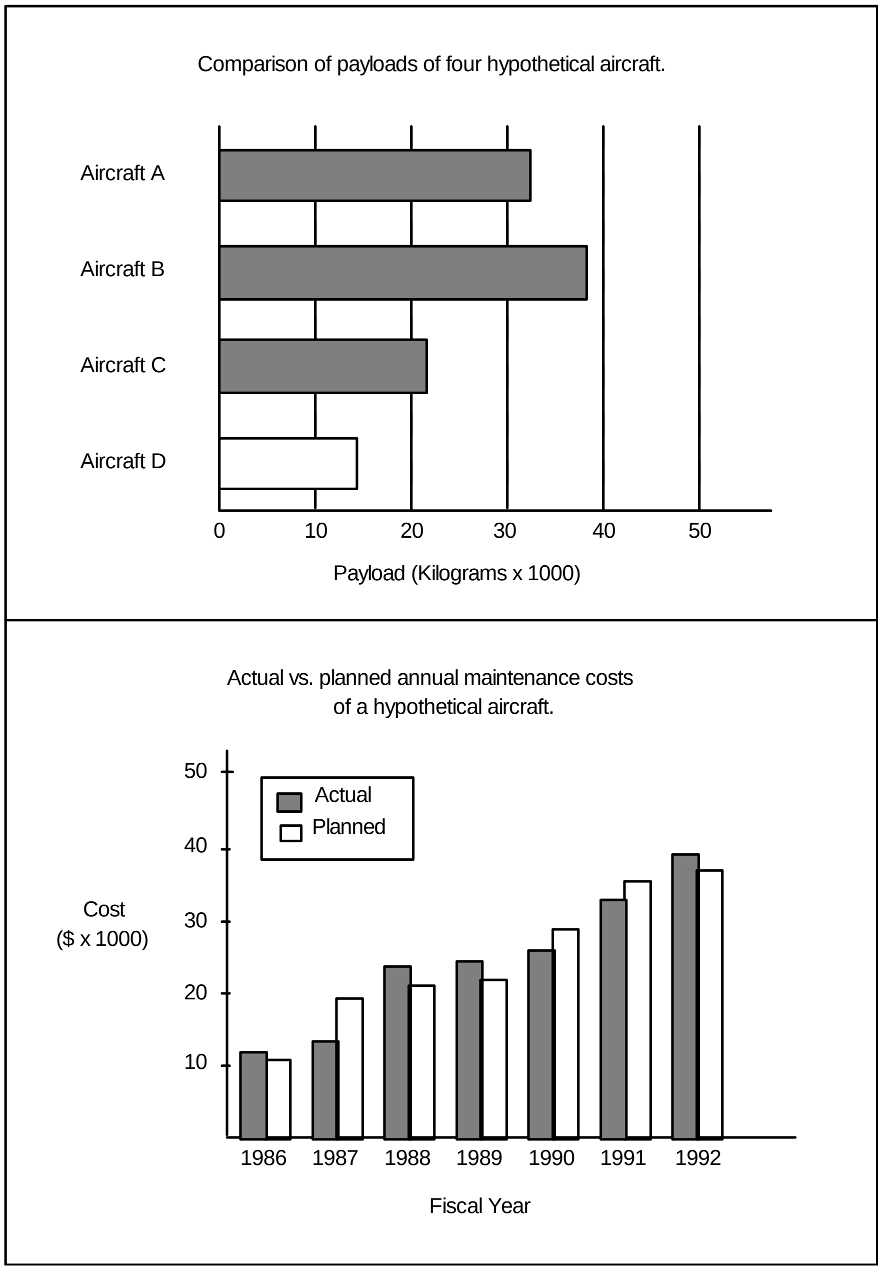


Figure 9-9. Example bar charts (from MIL-HDBK-761A).

If the number of bars being displayed is small, a bar graph should be used, with the bars separated, using one-half or less of the bar width as the spacing between bars. If the number of bars is large, a histogram should be used (i.e., eliminate the spacing between the bars). Coding (e.g., color, shading, texture) should be used to distinguish among different groups of bars or to highlight important data in one or more of the bars. If multiple bar graphs or histograms are presented, related groups of bars should be presented in a consistent order in each one. Each bar should be identified with its own text label, rather than presenting the labels in a separate legend.

The bar chart or histogram should be designed to conform to user expectations. For example, the use of icons (e.g., a silhouette of a person to represent 1,000 people) to represent quantitative information should be avoided. Charts and axes should be clearly labeled, and important information should be highlighted. When bars are presented in pairs, they should be labeled as a unit, with a legend provided that distinguishes between the bars.

Stacked bars should be used when both the total measures and the portions represented by segments are of interest. This arrangement of bars is similar to the surface chart. If stacked bars are used, the data categories should be presented in the same sequence. As with surface charts, data categories should be ordered so that the least variable are at the bottom of the bar and the most variable are at the top. The areas within each bar should be coded using different colors, shadings, or textures and identified by a text label displayed within the area.

9.5.1.3 Flow Charts

Applications should use a flow chart for presenting a schematic representation of sequences or processes. The path indicated in the flow chart should be left to right, top to bottom, or clockwise. Each decision point in the flow chart should contain a single, simple decision, as shown in figure 9-10. The elements and lines should be coded (e.g., symbol and shape coding) to assist in understanding, and the same coding scheme should be used throughout the flow chart. For example, the flow chart should provide directional indicators (arrows) to indicate the sequence to be followed. A legend should be included that describes each element and code used in the flow chart, and critical information and/or steps should be highlighted.

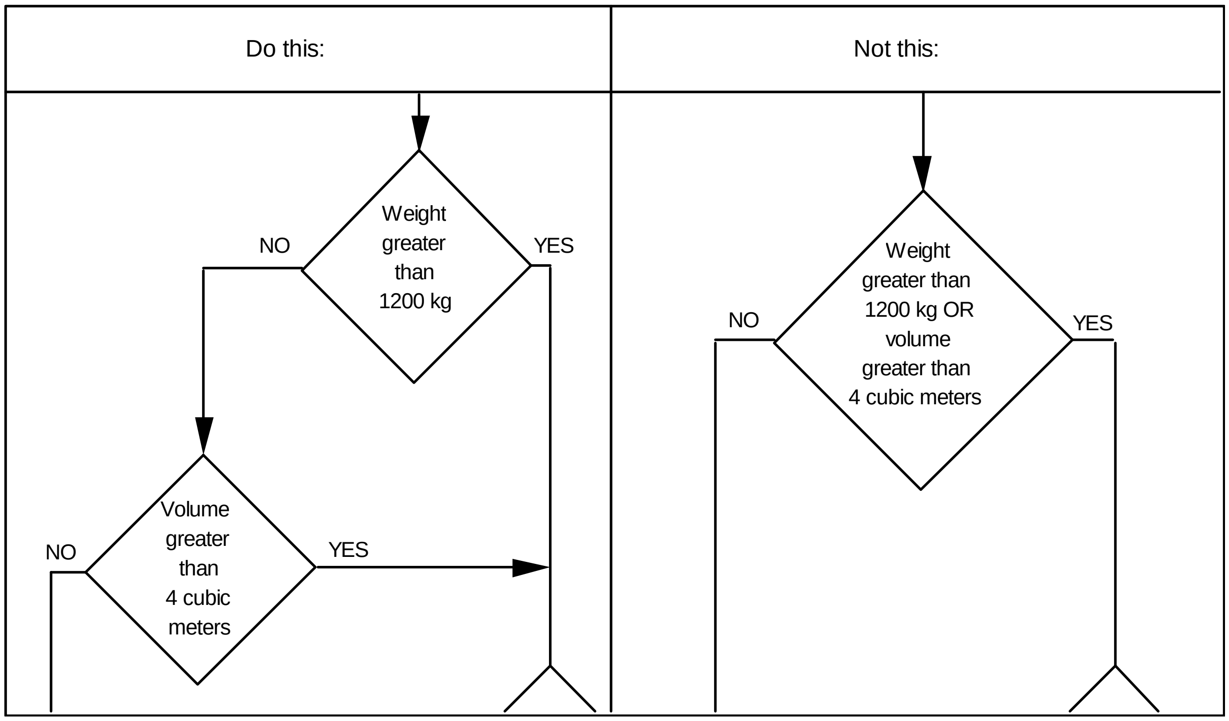

Do this:
Not this:
Weight greater than 1200 kg
NO
YES
Volume greater than 4 cubic meters
NO
YES
Weight greater than 1200 kg OR volume greater than 4 cubic meters
NO
YES

Figure 9-10. Example of labeling decision points in a flow chart (from MIL-HDBK-761A).

The steps presented in a flow chart should be ordered logically (i.e., in the sequence of operations, steps, or processes from start to finish), or the most important decisions or decisions that can be made with the greatest certainty should be placed first. If no ordering scheme can be identified, the flow chart should be organized to minimize the length of the path through it.

The shapes (e.g., boxes) used in the flow chart should follow existing shape coding conventions, and the text presented in the chart should be oriented for normal reading. Important elements (e.g., paths through the chart) should be emphasized through coding such as color.

9.5.1.4 Pie Charts

Applications should use a pie chart (shown in figure 9-11) to provide an approximation of how an entity is apportioned into component parts. If an accurate estimate of proportions or quantitative information is needed, a bar chart should be used instead of a pie chart.

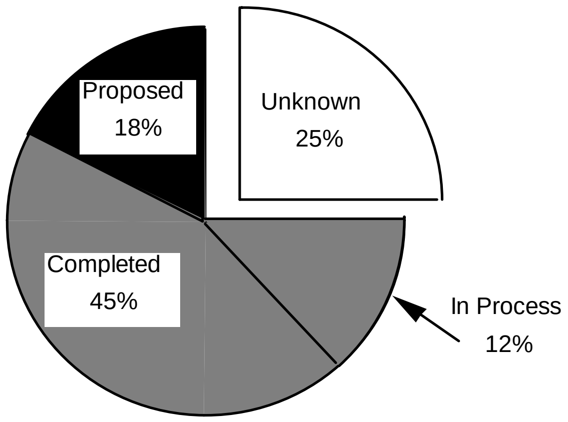


Figure 9-11. Example pie chart (from the DoD style guide).

The circle presented in a pie chart should be divided into five segments or less, and each segment should take up at least 5 percent (18 degrees) of the circle. Each segment should be coded using different colors, shadings, or textures and identified by a text label (presented in normal orientation) within the area. If the segment is too small to contain the label, it should be placed outside the segment, with a line from it to the segment. The label should describe the content of the segment and include the number (i.e., percentage or actual value) being represented by the segment. Segments can be emphasized by using special shading and by displacing them slightly from the remainder of the pie chart.

9.5.2 Manipulation of Graphical Data

It is expected that where appropriate, systems will provide the full range of graphic manipulation capabilities through the use of COTS software. When an application allows users to manipulate graphic data, it should provide the basic capabilities described in this section.

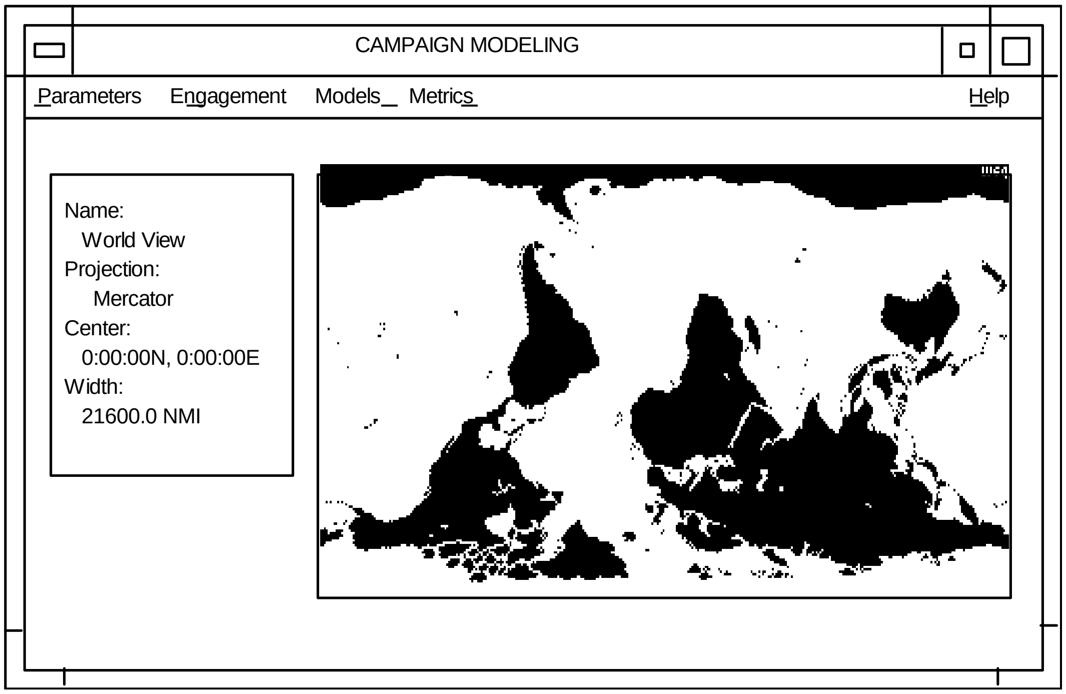
Users should be able to select and edit the attributes of objects (e.g., color, line thickness, font size), change object sizes (enlarge or reduce), and fill the enclosed area with colors or patterns. Applications should automatically align objects to an invisible rule or grid structure, complete figures (e.g., closure of a polygon), and draw lines between user-specified points. Object selection and transfer methods (e.g., drag and drop) should be available and implemented according to specifications presented in sections 3.3 and 3.4. Applications should provide a palette of drawing tools (as in figure 8-3) as part of the graphics display window to support efficient manipulation of objects by users.

Users should be able to draw objects such as lines, rectangles, ovals, and arcs; these objects should emerge as they are drawn and be easy to reposition, duplicate, and delete. Users should be able to group separate objects into a single object (e.g., combine text and a rectangle into a complex object). The objects that are displayed in the window should either be formatted to reflect how they will appear when printed (i.e., what you see is what you get), or users should be provided with an option to display the objects in this format before deciding to print them. A copy of the original graphics should be retained until users confirm that the objects are to be changed; the objects should not be modified automatically as users change them.

9.6 MAP WINDOWS

9.6.1 Window Design

A map window may contain both the map display and a set of controls for manipulating the map, or the entire window may be devoted solely to the map, with map controls available in individual dialog windows. A map window should include identifying information about the map (e.g., map name, coordinates, area, scale) along with status information (e.g., "drawing map"). This information may be presented in a subarea within the window (as in figure 9-12) or may be included along the bottom margin of the window. A continuous coordinate indicator giving the pointer location on the map should be available in a standard area of the window (e.g., with the identifying information at the bottom of the map).



Name:
World View
Projection:
Mercator
Center:
0:00:00N, 0:00:00E
Width:
21600.0 NMI

Figure 9-12. Example map window with identifying information (from the JMCIS style guide).

Each map should be displayed using the same orientation (i.e., north toward the top of the window), and the important features should be labeled. The labels should be positioned consistently with respect to the feature they describe (e.g., next to or below the feature), should not obscure important information or clutter the map, and should remain legible at all map resolutions.

When users display a color overlay on a map, a color coding key should also be displayed (e.g., in a subarea of the map window or in a separate dialog window) so that users can interpret the information in the overlay. Users should be able to display the coding key as desired without having to redisplay the overlay. If the coding key is presented in a dialog window, the window should be the minimum size needed to present the required information and positioned so that it obscures as little of the information on the map as possible. If appropriate, the key should function as a scale so that users can interpret the coding in the overlay easily and accurately.

Standard symbology[63] and color codes (see section 8.3.2) should be employed, with help available to users for identifying unknown symbols or other map information. If

[63]. The DoD style guide recommends that map graphic symbols conform with published standards such as NATO Standardization Agreement 2019 Military Symbols for Land-Based Systems, Army Field Manual 101-5-1 Operational Terms and Symbols, and the DIA Standard Military Graphics Symbols Manual. MIL-STD 2525 defines a common warfighting symbology for C^4I systems fielded by all of the military services. Implementation of the symbology is recommended in version 1 of the document but will be mandatory when version 2 is published in 1996.[64]. Each color's luminance and its CIE (International Commission on Illumination) chromaticity coordinates will be specified

multiple symbol sets are available, users should be able to select the set desired and to switch between sets without losing data. Applications that create new symbology should use shape coding in ways that are consistent with standard tactical symbology (e.g., pointed shapes are “hostile,” round or curved shapes are “friendly”) and match user expectations. Symbols should be placed on the map accurately or connected to the desired location using arrows, lines, or other graphics. Symbol labels should appear next to the symbol and present essential information (e.g., unit or track identification) about the symbol. The background of the symbol and label should be transparent so as not obscure other information (such as overlays) displayed on the map.

9.6.2 Map Manipulation

Users should be able to customize a map window to fit the task being performed. They should be able to pan and zoom a map as desired. A position or change indicator should be provided as a means for users to return quickly to the normal or starting map. When a map is zoomed, the size of symbols, labels, and other map features should be adjusted so that users can read them. In addition, users should be able to define a baseline (i.e., home) position on a map and be able to return to this position quickly (e.g., by selecting the appropriate menu item). Users should be able to place the window where desired, define the appearance of the map (e.g., select country colors), select the objects that will appear on the map (e.g., tactical overlays, contact data), and change the appearance of critical tactical information on the map (e.g., the color of friendly/hostile/neutral contacts).

Users should also be able to add, edit, reposition, and delete labels and overlays on the map and determine the distance and bearing between any two points on the map. Other recommended functions that should be available to users are areal computation/verification, area of interest selection, and area bounding boxes. Latitude and longitude should be enterable to the level of accuracy needed (e.g, degrees, minutes, seconds, tenths or hundredths of seconds); when calculations such as range, bearing, and position are performed, the answer computed should reflect the degree of accuracy appropriate to the scale of the map displayed. Graphics (e.g., an overlay or symbol) drawn on the map should include visual indications (i.e., “handles” displayed on the object) to define the portion of the graphic that is selectable and show where the graphic will be positioned when it is moved to a new location.

9.6.3 Symbology Manipulation

Users should be able to select and deselect map symbology by following the methods described in table 9-2. If additional selection methods are available, they shall be executed in a manner that complies with the specifications presented here; e.g., the pointer shall indicate the locus where pointing device operations occur (see section 2.2.1). Users should also be able to identify and select one or more tracks from the keyboard (e.g., by executing a database search of stored track information, with tracks that match the search criteria highlighted on the map). If symbols are densely packed or closely overlapping, users should be able to quickly and accurately select the symbol desired (e.g., by selecting from a pop-up list containing all of the symbol labels).

Table 9-2. Pointing device selection methods in a map window.

To select one track:
Position the pointer on a track (symbol or label) and click the Select button on the pointing device. If previously unselected, the track is selected (and highlights); any tracks that were already selected remain highlighted.

To select multiple tracks one at a time:
Position the pointer on a track and click the Select button to select it. Add tracks to the selection by clicking on other unselected tracks. Each track highlights as it is selected.

To select a group of contiguous tracks:
Position the pointer near the first track in the range to be selected, then press the Select button on the pointing device to set the anchor for the range. Drag the pointer until it is beyond the last track in the range, and release the button to complete the selection. As the pointer is dragged over the tracks, a bounding box is displayed outlining the tracks being selected. When the Select button is released, the box disappears and the tracks that were in the box are selected (and highlight).

To deselect one track:
Position the pointer on a track and hold down <Shift> while clicking the Select button on the pointing device. The track is deselected and returns to its normal appearance.

To deselect all tracks:
Position the pointer on an empty part of the map, and double click the Select button on the pointing device. All previously selected tracks are deselected and return to their normal appearance.

Users should be able to view or declutter overlapping symbol labels and obtain additional information, including exact map coordinates, for selected symbols (e.g., by double clicking on the symbol). The intensity of the map should be adjustable so that selected portions of the map can be faded out without losing all map features. Users should be able to distinguish among symbols that represent coincident points and to obtain information that will allow them to resolve ambiguities among the symbols.

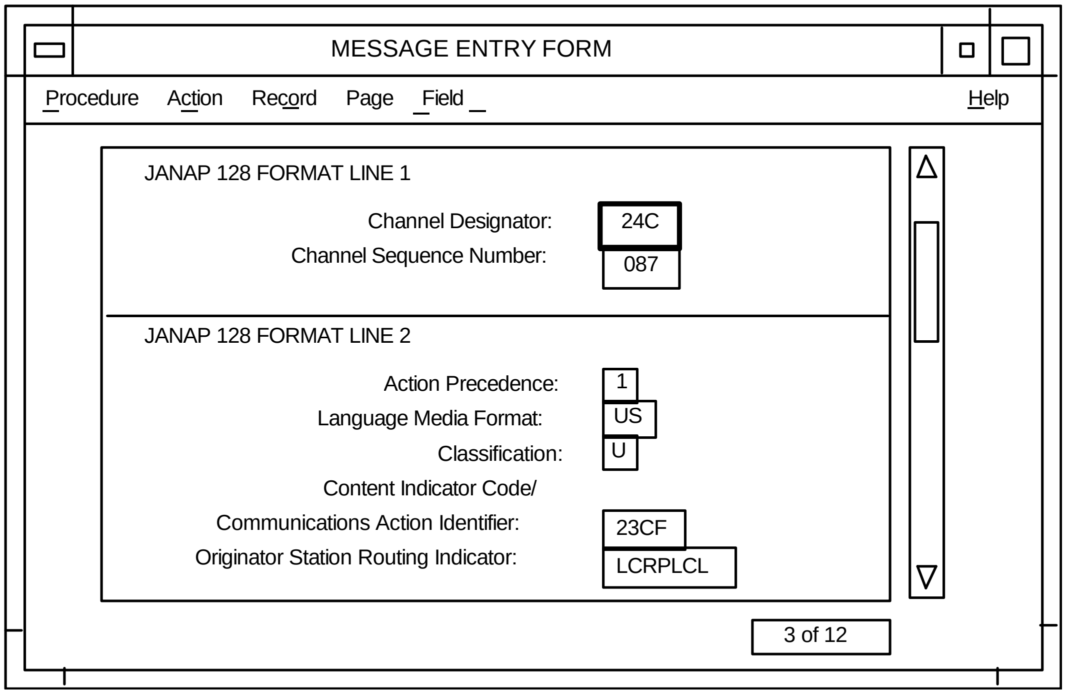
9.6.3 Automatic Updating

Users should be able to select the categories of tactical information that will be automatically updated and specify the frequency and rate at which the information is updated. If appropriate to the application, users should be able to temporarily stop and then resume further updates.

9.7 MESSAGE HANDLING WINDOWS

9.7.1 Message Preparation

Windows supporting the preparation of military messages, such as the one shown in figure 9-13, should follow the same general design as data entry windows, and users should be able to enter message information using the same data entry procedures.



JANAP 128 FORMAT LINE 1

Channel Designator:	24C
Channel Sequence Number:	087

JANAP 128 FORMAT LINE 2

Action Precedence:	1
Language Media Format:	US
Classification:	U
Content Indicator Code/ Communications Action Identifier:	23CF
Originator Station Routing Indicator:	LCRPLCL

Figure 9-13. Example message preparation window (from the JMCIS style guide).

Users should be provided with a basic set of message header fields and should be supported whenever possible in specifying the message address. For example, option menus should be available for selecting from limited sets of frequently used terms (e.g., classification level), and when replying to a message, the appropriate addressee(s) should be provided automatically. Users should be able to build and maintain distribution lists of commonly used addresses and select from these lists (without having to reenter the information) when addressing messages. The address should be checked for accuracy prior to transmission, and users should have to correct any errors before the message is sent.

Preformatted forms that conform to standard message format should be available; when users enter the text of a message, format control should be automatic. Users should be able to specify the data to be transmitted in the message, incorporate existing file data (including other messages received or transmitted) if desired, and save the message during preparation and upon completion.

9.7.2 Message Transmission

Message transmission procedures should be designed to minimize the user actions required. Users should be able to initiate message transmission directly (e.g., by selecting a Transmit push button). If a message cannot be sent immediately, it should be queued automatically so that users do not have to actively monitor the transmission and undelivered messages will be saved in the event of transmission failure. Users should be able to assign message priorities and to cancel or abort a transmission that has not been completed. Feedback should be available on status of message transmission, confirming that messages have been sent and indicating when transmission failures occur. Users should be able to specify what feedback they want to receive, and an automatic log of this information should be maintained.

9.7.3 Message Receipt

Users should be informed when high priority messages are received. For example, during login, a system may provide users with a list of new messages received since they last accessed the system. In addition, during a session, a system may activate an alert indicator in the system window to inform users of priority messages. Message notification should not interfere with ongoing system use but should provide some indication of urgency if the messages have different degrees of priority.

Incoming messages should be automatically queued by time of receipt and message priority, and logs of this information should be maintained. Users should be able to review summary information on messages that have been queued, display individual messages, save/file the ones of interest, and discard those that are unwanted. When a message is displayed, it should appear in a text window, and users should be able to scroll, save, and print it as they would any other text document.

9.8 IMAGERY WINDOWS

9.8.1 Window Design

An imagery window should contain an imagery display area and a set of tools for manipulating the image. These tools may be presented in the form of a palette or available as sets of controls grouped according to the nature of the operation performed. As in map windows, the imagery window should include identifying information about the image currently displayed, presented in a standard location (e.g., the message bar) in the window. The window should provide options (e.g., vertical and horizontal scroll bars, roam and zoom functions) for viewing the content of the display area when entire image does not fit in the window.

Users should be able to access and retrieve an image for display from a directory of images. Users can either specify the name of the image or search the directory for images matching user-defined criteria, including wildcard searches. Users should be able to print (in full-resolution or low-resolution) the image currently displayed in the window.

9.8.2 Basic Viewing Functions

An imagery window should provide access to a number of basic image exploitation functions. Users should be able to roam the image (similar to panning in a map window) in both horizontal and vertical directions. Automatic, jump, manual, and patterned roam control should be available, with users specifying the rate, direction, and area of interest as appropriate. As users roam an image, they should be able to tag specific areas of interest for later recall while the image remains in the display area. Users should also be able to zoom an image, either in predefined steps or in increments they specify. An imagery window should support "image chipping" (i.e., allow users to select and designate regions of an image for storage). Finally, users should be able to create and manipulate annotations to display with an image. Annotations include text, lines, icons, geometric shapes, colors, and patterns. In each case, users should be able to edit, delete, reposition, resize, save, and retrieve the annotation without altering the underlying imagery data. In addition, when users manipulate (e.g., rescale) the image, the scale and orientation of the annotations should be adjusted to accommodate the changes.

9.8.3 Advanced Viewing Functions

An imagery window should support imagery-unique operations, including image enhancement filtering capabilities and allowing users to control the dynamic range of image data by modification of intensity and contrast. Users should be able to select and then examine (e.g., zoom, roam) a region of interest (ROI) within an image, leaving the remainder of the image unaffected. If desired, users should be able to apply the ROI to the entire image. An imagery window should provide mensuration functions (for computing lengths, areas, and volumes from dimensions or angles) and perform isotropic pixel correction (i.e., convert rectangular pixels to square pixels for display purposes). Finally, users should be able to create a mirror view of the image so they can adjust the image if the negative was inverted when scanned.

An imagery window should provide advanced functions for viewing geo-referenced images. The window should provide a default image rotation where "up is up" (i.e., vertical objects are oriented toward the top of the window) and an automatic north rotation (i.e., with North orientated at the top of the window), as well as allow interactive rotation in user-defined increments. In addition, annotation capabilities should be extended so that when users manipulate the image, the scale and orientation of the annotations are adjusted according to the type or nature of the symbol (e.g., icons do not change size but text does).

An imagery window should allow users to plot user-selected geographic data on the image, including frame-by-frame plotting to animate the data in either a forward or reverse time direction. With respect to image fusion operations, users should be able to register (i.e., transform an image so that it aligns with either another image or a map projection) geo-referenced images acquired from the same or different sensors and display them together.

Users should be able to perform concurrent geometric manipulations on separate geo-referenced windows that have overlapping geographic coverage. They should be able to "slave" together multiple geo-referenced windows and manipulate (e.g., roam, zoom, rotate) all the slaved windows concurrently and relatively (i.e., with the window centers maintained at a common center lat/long position, despite differences in the amount and distance coverage between windows).

9.9 BRIEFING SUPPORT WINDOWS

9.9.1 Window Design

A briefing support window should contain a display area for viewing and editing the slides in a briefing, and a set of text, drawing, and graphing tools in the form of a palette to create and manipulate objects on a slide (e.g., as in figure 8-3). The window may also include an icon bar for quick access to commonly used commands for slide manipulation (e.g., to select font and color scheme, to change view scales). The window should be titled with the name of the briefing and include a standard location (e.g., the message bar of the window) for indicating the current slide number within the briefing. The window should provide vertical and horizontal scroll bars for viewing the content of the display area when entire slide does not fit in the window.

9.9.2 Building Briefing Slides

A briefing support window should provide users with a full range of graphic and text manipulation capabilities as described in sections 9.3, 9.4, and 9.5. Graphic manipulation features include the ability to edit objects (e.g., cut, copy, paste, duplicate, delete), change the attributes of an object (e.g., fill, line, shadow) and the order of objects in a stack, move an object by dragging it, and group objects together to create another object. Word processing features include access to multiple fonts and type sizes, spellchecking, and search/replace. Users should be able to select the color schemes used in each slide, including the colors for the background (the color that appears behind all the other objects), lines and text, shadows, title text (the color of text for the slide title), fills (the color that appears inside an object), and accents (the colors used to emphasize special areas on a slide).

When creating a set of briefing slides, users should be able to specify global features (e.g., font, color scheme, graphic template) for the briefing as a whole and modify these settings for individual slides as desired. Users should be able to import text and graphics from other sources, incorporate clip art, animation, and video as part of the briefing, and do screen captures (e.g., from the tactical display) and annotate them with text and other graphics as overlays.

Users should be able to create a file containing each set of slides (i.e., briefing). The file content should include the individual slides in the briefing as well as the order in which they are displayed. Users should be able to reorder, add, and delete slides and then save the result; they should also be able to select and copy multiple slides to re-use in the current briefing or to paste into another briefing. Users should be able to access, retrieve, and rename the briefings they have created as well as print some or all of the slides in a briefing.

9.9.3 Presentation Management

Briefing support windows should include an option for presenting a slide show (i.e., displaying the slides in a briefing so that they fill the screen and all of the tools, menus, and other screen elements are hidden). Users should be able to select all or some of the slides for presentation as well as specify the slide advance, sequence, timing, and transitions for the presentation. When the slides are displayed in a presentation, all of them should have the same output size, aspect ratio, and orientation (i.e., all horizontal or all vertical). As users give the presentation, they should be able to use the keyboard or the pointing device to move between slides, stop and restart an automatic show, temporarily blank the screen, and end the show. If the pointer remains visible during the presentation, users should be able to move it with the pointing device and use it to point to and draw on the slides.

9.10 GRAPHICAL SCHEDULING WINDOWS

9.10.1 Window Design

A graphical scheduling window should be used to display timelines or scheduled events. The schedule should be displayed with time presented on the horizontal axis and the tasks to be performed arrayed vertically, as shown in figure 9-14. The window should be titled with the schedule name and include vertical and horizontal scroll bars when entire schedule does not fit in the window.

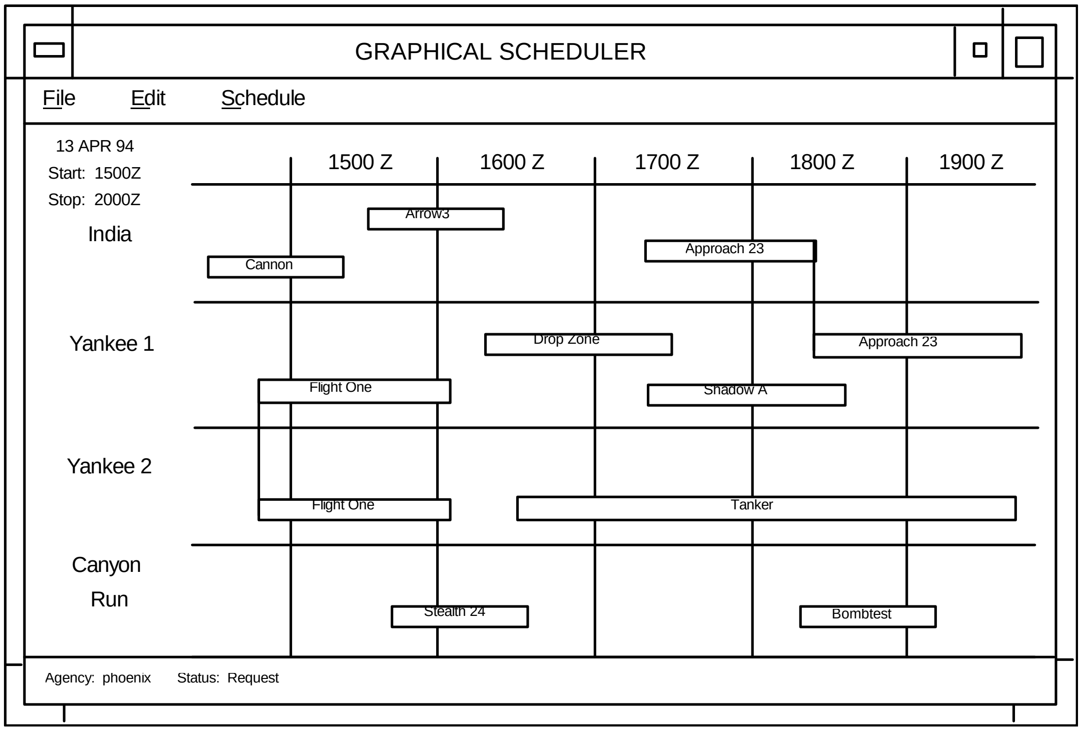


Figure 9-14. Example graphical scheduling window (from the TBM style guide).

9.10.2 Window Content

Each event in a schedule should be represented by an event icon (in the form of a line or bar) whose length is proportional to the amount of time necessary to complete a task. The icon should be displayed to the right of its associated task. Figure 9-15 provides an example of task and event labels and event icons. If different types of events (e.g., ones undertaken at different locations) are presented on a schedule, they should be differentiated by color or shading or include an alphanumeric designator displayed on or above the icon for the event. If a coding scheme is applied to the schedule, users should be able to access a legend or key that describes the coding technique used. No more than nine uniquely coded event icons should be presented on a schedule at one time.

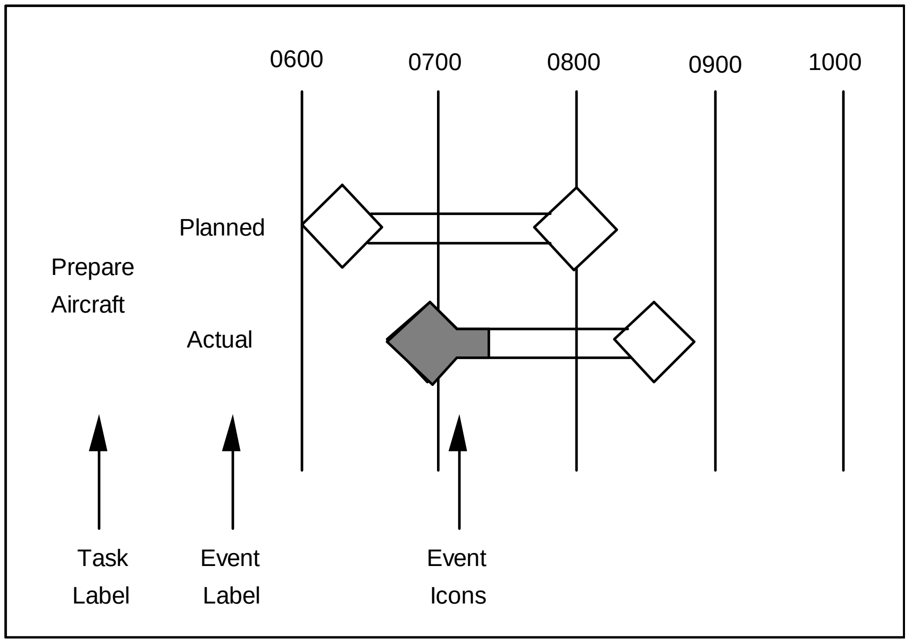


Figure 9-15. Example task and event labels and event icons (from the TBM style guide).

If more than one event icon is used per task, each icon should be labeled. For example, a schedule might include event icons representing planned and actual times, or earliest, latest, and actual times. Event icon labels should be placed along the vertical axis or on or above the timeline. If appropriate, different scheduling attributes can be represented by displaying symbols with event icons. These symbols can be formed from various geometric shapes (e.g., circles, diamonds, squares) and fill patterns (e.g., filled symbols for events that are underway and hollow symbols for future events) should be used to indicate various schedule situations.

If the schedule is cluttered or users require a high degree of precision, gridlines may be used to improve the ease and accuracy with which to read information on the schedule. A gridline should be displayed to indicate the present date and time; users should be able to display or hide this line as required.

9.10.3 Schedule Manipulation

Users should be able to define the start and stop time of the schedule displayed in the window and to do so to the desired degree of precision (e.g., day, hour). Schedule duration can be a superset of what can be displayed in the window at one time. Users should also be able to display all or a portion of the preselected duration time. For example, users may choose to display only selected days from a schedule with a one-week duration.

Users should be able to select an individual event icon and obtain additional information about that event. The selection methods available to users should be implemented in accordance with the specifications provided in this style guide.

If frequent schedule changes are anticipated, users should be able to reschedule an event icon by directly manipulating the event icon using the object transfer methods described in this style guide. If exact positioning of an event icon with the pointing device is difficult, the application should provide alternative methods for users to locate the icon. For example, the application might allow users to enter the start and stop times for the event using text fields presented in a dialog window or provide a user definable grid that automatically repositions event icons after users place them in a new location.

APPENDIX A
RESOURCE SETTINGS

This appendix provides the resource settings that implement the “look and feel” called for by the specifications presented in this document. Developers shall build their applications to comply with the resource configuration presented here. If developers create new objects or modify existing ones to provide the functionality desired, the appearance of these objects should be consistent with the resource settings provided in this appendix.

Table A-1 contains resource settings for the Motif window manager, and table A-2 provides them for the window components and controls listed in table 8-1. Because the tables are provided as a reference for developers, some of the settings are the Motif default for the resource; it is understood that these settings would not be included in an executable file. If specific colors are entered for the top and bottom shadow colors (rather than using the Motif default which sets them dynamically), the colors selected should match those generated when the Motif color algorithm is used. In the event of discrepancies between the resources listed here and the specifications presented in this document, the specifications (e.g., tables 7-1 and 8-1) shall take precedence.

The settings in tables A-1 and A-2 were originally derived from information in the OSF/Motif Programmer's Reference and Motif Reference Manual. In addition, recommendations provided by Kobara in Visual Design with OSF/Motif are included where they are consistent with the specifications contained in this document. Developers should consult this reference for guidance on how to build a user interface that is consistent with Motif design philosophy.

A.1 Resource Settings for Unified Build Applications

The user interface of the Unified Build applications included in GCCS was originally developed using X Window and the Wizard graphical user interface toolkit. These applications have been modified so that they now make calls to Motif in order to deliver a user interface in accordance with the standards profile defined in the TAFIM. While the conversion process was designed to produce a style that was as Motif-compliant as possible, there are some Motif features that are not yet duplicated exactly by the conversion process. In addition, because some of the functionality in Unified Build can be provided only through the Wizard toolkit, the user interface for these applications retains these Wizard components so that there would be no loss in functionality as a result of the conversion process. The intent is to provide this functionality in Motif as soon as the necessary capabilities are available in that toolkit.

Developers should be aware that the following user interface features in Unified Build applications differ from the appearance and behavior defined in the GCCS style guide.

a. Unified Build applications use the Wizard scrolled list widget in order to support color coding of list items. The colors assigned to these items have not been changed from the previous Wizard implementation.

b. Some of the underlying features of the Wizard toolkit have been retained in the Motif conversion and impact the “look and feel” of the user interface of Unified Build applications. For example, text fields have a two-dimensional, rather than three-dimensional, appearance in order to accommodate the original spacing of controls in windows. In addition, the manner in which tab groups are defined in Wizard differs from that used in Motif so that keyboard navigation in Unified Build windows may diverge from the guidelines provided in this style guide.

c. User interface behavior in windows containing widgets from both toolkits depends on the position of the pointer. When the pointer is on a Wizard object (i.e., a scrolled list), the behavior follows Wizard; when the pointer is on a Motif object, the behavior follows Motif. For example, because location cursor behavior is defined differently in the two toolkits, moving the pointer between Wizard and Motif objects alters this behavior.

d. In order to field a single version of Unified Build that supports both versions 1.1 and 1.2 of Motif, text entry mode was hardcoded so that users have access only to replace mode. When the software moves to version 1.2 of Motif on all platforms, the ability to toggle between insert and replace modes will be made available. In addition, the option to set the <Insert> toggle at other than a field level will be addressed at that time.

Table A-3 lists resource settings that produce a Motif "look and feel" for window components and controls in applications developed with the Wizard toolkit. Applications that are based on the standard Wizard APIs should be able to use the Motif conversion of Unified Build without modification. Applications that bypass the standard Wizard APIs will require modification in order to use the Motif conversion of Unified Build.

A.2 Color Names and RGB Values

The color specifications presented in this document require that developers adhere to the RGB values indicated below. In some cases, however, there may be hardware limitations that allow only 256 colors to be allocated so that it is not possible to allocate the requested colors. It is recommended that applications that allocate colors ensure that color was actually allocated. If not, the application should determine the closest match. Doing so will ensure that the color set required in the style guide is, in fact, executed (via use of the closest match), even if the color requested is beyond the number of colors available in the color table.

The following colors (and RGB values) are called out in this document:[64]

System Window		Application Windows, Menus, and Controls	
Colors TBD		Colors TBD	
Classification Markings		Coding of Tactical Information	
Green	TBD	Red	TBD
Blue	TBD	Green	TBD
Red	TBD	Yellow	TBD
Orange TBD	Blue	TBD	
		Purple	TBD

A.3 Replace Mode for Text Entry

The Text and Text Field widgets in version 1.2 of Motif provide support for replace mode, but it is not bound by default to <Insert>. The binding can be done by adding the following lines of code to the .Xdefaults file:

```
*XmText.translations:  #override\n\
        :<Key>osfInsert:  toggle-overstrike()
*XmTextField.translations:  #override\n\
        :<Key>osfInsert:  toggle-overstrike()
```

Table A-1. Resources and settings for the Motif window manager.

Resource	Setting
Appearance of Window Components	
Mwm*background:	TBD
Mwm*bottomShadowColor:	Default (Dynamic)
Mwm*foreground:	TBD
Mwm*topShadowColor:	Default (Dynamic)
Mwm*activeBackground:	TBD
Mwm*activeBottomShadowColor:	Default (Dynamic)
Mwm*activeForeground:	TBD
Mwm*activeTopShadowColor:	Default (Dynamic)
Mwm*feedback*background:	TBD
Mwm*feedback*foreground:	TBD
Mwm*menu*background:	TBD
Mwm*menu*foreground:	TBD
MenuExec*background:	TBD
MenuExec*foreground:	TBD
Mwm*icon*background:	TBD
Mwm*icon*foreground:	TBD
Mwm*icon*activeBackground:	TBD
Mwm*icon*activeForeground:	TBD
Window Manager Behavior	
Mwm*autoKeyFocus:	True
Mwm*buttonBindings:	Default (“DefaultButtonBindings”)
Mwm*cleanText:	Default (True)
Mwm*clientAutoPlace:	False
Mwm*clientDecoration:	All
Mwm*colormapFocusPolicy:	Default (Keyboard)
Mwm*deiconifyKeyFocus:	Default (True)
Mwm*enforceKeyFocus:	Default (True)
Mwm*fadeNormalIcon:	True
Mwm*focusAutoRaise:	False
Mwm*frameBorderWidth:	6 (per Kobara)
Mwm*iconAutoPlace:	Default (True)
Mwm*iconClick:	Default (True)
Mwm*iconDecoration:	Default (label image)
Mwm*iconPlacement:	Default (left / bottom)

Note: Mwm*iconImageMaximum is not included in this table. If the following resource:

Mwm*iconImageMaximum:	Default

is included in the X.defaults file, the icons displayed on the screen will be much smaller than the default size. If this resource is omitted from the file, the icons will be displayed in the default size (i.e., 50 x 50 pixels).

Table A-1. Resources and settings for the Motif window manager (continued).

Resource	Setting
Mwm*interactivePlacement:	Default (False)
Mwm*keyBindings:	Default (“DefaultKeyBindings”)
Mwm*keyboardFocusPolicy:	Default (explicit)
Mwm*limitResize:	Default (True)
Mwm*lowerOnIconify:	False
Mwm*moveThreshold:	Default (4)
Mwm*multiScreen:	True
Mwm*passButtons:	True
Mwm*passSelectButton:	Default (True)
Mwm*positionIsFrame:	False
Mwm*positionOnScreen:	False
Mwm*raiseKeyFocus:	Default (False)
Mwm*resizeBorderWidth:	7 (per Kobara)
Mwm*resizeCursors:	Default (True)
Mwm*showFeedback:	Default (all)
Mwm*startupKeyFocus:	Default (True)
Mwm*traversalOn:	Default (True)
Mwm*useClientIcon:	Default (False)
Mwm*useIconBox:	Default (False)
Mwm*windowMenu:	Default (“DefaultWindowMenu”)
Mwm*wMenuButtonClick:	Default (True)
Mwm*wMenuButtonClick2:	Default (True)

Table A-2. Resources and settings for Motif widgets.

Resource	Setting
General Settings	
*background:	TBD
*cursor:	left_ptr
*fontList:	9x15bold, 8x13bold recommended
*foreground:	TBD
*highlightColor:	TBD
*pointerColor:	TBD
*traversalOn:	True
ArrowButton	
XmNbackground:	TBD
XmNbottomShadowColor:	Default (Dynamic)
XmNheight:	16 recommended (per Kobara)
XmNhighlightThickness:	Default (2)
XmNshadowThickness:	Default (2)
XmNtopShadowColor:	Default (Dynamic)
XmNwidth:	16 recommended (per Kobara)
Bulletin Board	
XmNmarginHeight:	2 recommended (per Kobara)
XmNmarginWidth:	2 recommended (per Kobara)
XmNallowOverlap:	False (per Kobara)
CascadeButton (in a Menubar)	
XmNbackground:	TBD
XmNbottomShadowColor:	Default (Dynamic)
XmNforeground:	TBD
XmNmarginHeight:	Default (2) recommended (per Kobara)
XmNmarginWidth:	8 recommended (per Kobara)
XmNmnemonic:	Not NULL
XmNshadowThickness:	Default (2)
XmNtopShadowColor:	Default (Dynamic)
CascadeButton (in a Pulldown or Popup Menu)	
XmNbackground:	TBD)
XmNbottomShadowColor:	Default (Dynamic)
XmNforeground:	TBD
XmNmappingDelay:	Default (180)
XmNmarginHeight:	Default (2) recommended (per Kobara)
XmNmnemonic:	Not NULL for pulldown menu
XmNshadowThickness:	Default (2)
XmNtopShadowColor:	Default (Dynamic)

Table A-2. Resources and settings for Motif widgets (continued).

Resource	Setting
Command Box	
XmNbackground:	TBD
XmNbottomShadowColor:	Default (Dynamic)
XmNforeground:	TBD
XmNhighlightColor:	TBD
XmNmarginHeight:	Default (10)
XmNmarginWidth:	Default (10)
XmNshadowThickness:	Default (Dynamic)
XmNshadowType:	Default (XmSHADOW_OUT)
XmNtopShadowColor:	Default (Dynamic)
DrawnButton	
XmNhighlightThickness:	2 if XmSHADOW_IN or XmSHADOW_OUT; Default (0) recommended otherwise
XmNshadowThickness:	Default (2) recommended (per Kobara)
XmNshadowType:	Default (XmSHADOW_ETCHED_IN) recommended when used as sortable column heading or in palette
File Selection Box	
XmNbackground:	TBD
XmNbottomShadowColor:	Default (Dynamic)
XmNforeground:	TBD
XmNhighlightColor:	TBD
XmNmarginHeight:	Default (10)
XmNmarginWidth:	Default (10)
XmNminimizeButtons:	Default(False)
XmNshadowThickness:	Default (Dynamic)
XmNshadowType:	Default (XmSHADOW_OUT)
XmNtopShadowColor:	Default (Dynamic)
Frame	
XmNbackground:	TBD
XmNbottomShadowColor:	Default (Dynamic)
XmNmarginHeight:	Not 0 (per Kobara)
XmNmarginWidth:	Not 0 (per Kobara)
XmNshadowThickness:	2 recommended for Shadow In and Shadow Out frames, 1 for Etched In and Etched Out frames (per Kobara)
XmNtopShadowColor:	Default (Dynamic)

Table A-2. Resources and settings for Motif widgets (continued).

Resource	Setting
Label	
XmNalignment:	XmALIGNMENT_BEGINNING for vertically stacked labels and labels with widgets to their left; XmALIGNMENT_END for labels with widgets to their right; otherwise XmALIGNMENT_CENTER recommended (per Kobara)
XmNbackground:	TBD
XmNforeground:	TBD
XmNmarginHeight:	6 recommended (per Kobara)
XmNmarginWidth:	Default (2) recommended (per Kobara)
XmNmnemonic:	Default (NULL)
XmNshadowThickness:	Default (0)
XmNtraversalOn:	Default (False)
List (also see Scrolled Window)	
XmNbackground:	TBD
XmNbottomShadowColor:	Default (Dynamic)
XmNforeground:	TBD
XmNhighlightThickness:	Default (2)
XmNlistMarginHeight:	2 recommended (per Kobara)
XmNlistMarginWidth:	2 recommended (per Kobara)
XmNlistSizePolicy:	XmCONSTANT recommended (per Kobara)
XmNlistSpacing:	Default (0)
XmNscrollBarDisplayPolicy:	XmSTATIC recommended (per Kobara)
XmNshadowThickness:	Default (2)
XmNtopShadowColor:	Default (Dynamic)
MessageBox	
XmNbackground:	TBD
XmNbottomShadowColor:	Default (Dynamic)
XmNforeground:	TBD
XmNhighlightColor:	TBD
XmNmarginHeight:	Default (10)
XmNmarginWidth:	Default (10)
XmNmessageAlignment:	Default (XmALIGNMENT_BEGINNING) recommended
XmNminimizeButtons:	Default (False)
XmNshadowThickness:	Default (Dynamic)
XmNshadowType:	Default (XmSHADOW_OUT)
XmNtopShadowColor:	Default (Dynamic)
PanedWindow	
XmNbackground:	TBD

XmNbottomShadowColor:	Default (Dynamic)

Table A-2. Resources and settings for Motif widgets (continued).

Resource	Setting
XmNmarginHeight:	Default (3)
XmNmarginWidth:	Default (3)
XmNsashHeight:	6 recommended (per Kobara)
XmNsashIndent:	Should equal the value by which vertical scrollbars are offset from the right edge of the paned window; -4 if no scrollbars are used (per Kobara)
XmNsashShadowThickness:	Default (Dynamic)
XmNsashWidth:	Should equal the width of the scrollbars in the application; 12 if no scrollbars used (per Kobara)
XmNseparatorOn:	Default (True)
XmNshadowThickness:	Default (2)
XmNspacing:	Default (8) recommended (per Kobara)
XmNtopShadowColor:	Default (Dynamic)
PushButton (in a Work Area)	
XmNalignment:	Default (XmALIGNMENT_CENTER)
XmNarmColor:	TBD
XmNbackground:	TBD
XmNbottomShadowColor:	Default (Dynamic)
XmNdefaultButtonShadowThickness:	Default (0)
XmNfillOnArm:	Default (True)
XmNforeground:	TBD
XmNhighlightThickness:	Default (2)
XmNmarginHeight:	Default (2) recommended (per Kobara)
XmNmarginWidth:	8 recommended (per Kobara)
XmNshadowThickness:	Default (2)
XmNshowAsDefault:	1 recommended for default push button (per Kobara); Default (0) otherwise
XmNtopShadowColor:	Default (Dynamic)
Push Button (in a Pulldown or Popup Menu)	
XmNbackground:	TBD
XmNbottomShadowColor:	Default (Dynamic)
XmNfillOnArm:	False
XmNforeground:	TBD
XmNhighlightThickness:	Default (2)
XmNmarginHeight:	Default (2) recommended (per Kobara)
XmNmarginWidth:	8 recommended (per Kobara)
XmNshadowThickness:	Default (2)
XmNtopShadowColor:	Default (Dynamic)
RowColumn	

XmNadjustLast:	False recommended (per Kobara)

Table A-2. Resources and settings for Motif widgets (continued).

Resource	Setting
RowColumn (for Menubar)	
XmNentryAlignment:	XmALIGNMENT_CENTER
XmNentryBorder:	2 recommended (per Kobara)
XmNorientation:	Default (XmHORIZONTAL)
XmNspacing:	Default (0) recommended (per Kobara)
RowColumn (for Pulldown, Popup, or Option Menu)	
XmNentryAlignment:	Default (XmALIGNMENT_BEGINNING)
XmNentryBorder:	Default (0)
XmNorientation:	Default (XmVERTICAL)
XmNspacing:	0 recommended (per Kobara)
Scale	
XmNbackground:	TBD
XmNbottomShadowColor:	Default (Dynamic)
XmNforeground:	TBD
XmNhighlightThickness:	Default (2)
XmNshadowThickness:	2 recommended
XmNshowValue:	True recommended
XmNtopShadowColor:	Default (Dynamic)
XmNtroughColor:	TBD
ScrollBar	
XmNbackground:	Should be the same as parent background color
XmNbottomShadowColor:	Default (Dynamic)
XmNhighlightThickness:	2 recommended (per Kobara)
XmNinitialDelay:	Default (250)
XmNrepeatDelay:	Default (50)
XmNscrollBarHeight:	16 recommended for horizontal scrollbars (per Kobara)
XmNscrollBarWidth:	16 recommended for vertical scrollbars (per Kobara)
XmNshadowThickness:	Default (2) recommended (per Kobara)
XmNshowArrows:	Default (True)
XmNtopShadowColor:	Default (Dynamic)
XmNtroughColor:	TBD
ScrolledWindow	
XmNscrollBarDisplayPolicy:	XmSTATIC recommended (per Kobara)
XmNscrollBarPlacement:	XmBOTTOM_RIGHT recommended (per Kobara)
XmNscrolledWindowMarginHeight:	Default (0) recommended for scrolled list; 2 recommended for scrolled text (per Kobara)

XmNscrolledWindowMarginWidth:	Default (0) recommended for scrolled list; 2 recommended for scrolled text (per Kobara)

Table A-2. Resources and settings for Motif widgets (continued).

Resource	Setting
XmNspacing:	Default (4) recommended (per Kobara)
Note: Use the following resource to set the trough color toTBD in a scrolled window:	
XmScrolledWindow.XmScrollBar.troughColor:	TBD
Selection Box	
XmNbackground:	TBD
XmNbottomShadowColor:	Default (Dynamic)
XmNforeground:	TBD
XmNhighlightColor:	TBD
XmNmarginHeight:	Default (10)
XmNmarginWidth:	Default (10)
XmNminimizeButtons:	Default (False)
XmNshadowThickness:	Default (Dynamic)
XmNshadowType:	Default (XmSHADOW_OUT)
XmNtopShadowColor:	Default (Dynamic)
Separator	
XmNbackground:	TBD
XmNmargin:	Default (0) recommended (per Kobara)
XmNseparatorType:	XmSHADOW_ETCHED_IN recommended (per Kobara)
XmNshadowThickness:	Default (2) recommended (per Kobara)
XmNtraversalOn:	Default (False)
Text (Editable)	
XmNautoShowCursorPosition:	Default (True)
XmNbackground:	TBD
XmNblinkRate:	Default (500)
XmNbottomShadowColor:	Default (Dynamic)
XmNcursorPositionVisible:	Default (True)
XmNeditable:	Default (True)
XmNforeground:	TBD
XmNhighlightThickness:	Default (2)
XmNshadowThickness:	Default (2)
XmNtopShadowColor:	Default (Dynamic)
XmNwordWrap:	True recommended (per Kobara)
Text (Noneditable)	
XmNautoShowCursorPosition:	False
XmNbackground:	TBD
XmNblinkRate:	0
XmNbottomShadowColor:	Default (Dynamic)

XmNcursorPositionVisible:	False

Table A-2. Resources and settings for Motif widgets (continued).

Resource	Setting
XmNeditable:	False
XmNforeground:	TBD
XmNshadowThickness:	Default (2)
XmNtopShadowColor:	Default (2)
XmNtraversalOn:	False
TextField (Containing Editable Text)	
XmNbackground:	TBD
XmNblinkRate:	Default (500)
XmNbottomShadowColor:	Default (Dynamic)
XmNcursorPositionVisible:	Default (True)
XmNeditable:	Default (True)
XmNforeground:	TBD
XmNhighlightThickness:	Default (2)
XmNmarginHeight:	2 recommended (per Kobara)
XmNmarginWidth:	Default (5) recommended
XmNshadowThickness:	Default (2)
XmNtopShadowColor:	Default (Dynamic)
TextField (Containing Noneditable Text)	
XmNbackground:	TBD
XmNblinkRate:	0
XmNbottomShadowColor:	Default (Dynamic)
XmNcursorPositionVisible:	False
XmNeditable:	False
XmNforeground:	TBD
XmNhighlightThickness:	0
XmNmarginHeight:	2 recommended (per Kobara)
XmNmarginWidth:	Default (5) recommended
XmNshadowThickness:	Default (2)
XmNtopShadowColor:	Default (Dynamic)
XmNtraversalOn:	False
ToggleButton (in a Work Area)	
XmNbackground:	TBD
XmNbottomShadowColor:	Default (Dynamic)
XmNfillOnSelect:	Default (True)
XmNforeground:	TBD
XmNhighlightThickness:	Default (2)
XmNindicatorOn:	Default (True)
XmNmarginHeight:	Default (2) recommended (per Kobara)
XmNmarginWidth:	Default (2) recommended (per Kobara)
XmNselectColor:	TBD
XmNshadowThickness:	Default (Dynamic)

XmNspacing:	Default (4) recommended (per Kobara)

Table A-2. Resources and settings for Motif widgets (continued).

Resource	Setting
XmNtopShadowColor:	Default (Dynamic)
XmNvisibleWhenOff:	Default (True)
ToggleButton (in a Pulldown Menu)	
XmNbackground:	TBD
XmNbottomShadowColor:	Default (Dynamic)
XmNfillOnSelect:	Default (True)
XmNforeground:	TBD
XmNindicatorOn:	Default (True)
XmNmarginHeight:	Default (2) recommended (per Kobara)
XmNmarginWidth:	Default (2) recommended (per Kobara)
XmNmnemonic:	Not NULL
XmNselectColor:	TBD
XmNshadowThickness:	Default (Dynamic)
XmNspacing:	Default (4) recommended (per Kobara)
XmNtopShadowColor:	Default (Dynamic)
XmNvisibleWhenOff:	Default (False)

Table A-3. Resources and settings for the Wizard toolkit.

Resource	Setting
General Resources	
WizardXm*background:	TBD
WizardXm*foreground:	TBD
WizardXm*highlightColor:	TBD
Buttons	
WizardXm*buttons.armColor:	TBD
WizardXm*buttons.background:	TBD
WizardXm*buttons.foreground:	TBD
Color Sort List	
WizardXm*XmSortList.background:	TBD
Error Warning Window	
WizardXm*Error.clientDecoration:	None
WizardXm*Error.background:	TBD
WizardXm*Error.foreground:	TBD
WizardXm*Error.highlightColor:	TBD
Label	
WizardXm*XmLabel.background:	TBD
WizardXm*XmLabel.foreground:	TBD
WizardXm*XmLabel.marginHeight:	0
WizardXm*XmLabel.marginWidth:	0
WizardXm*InactiveLabel.background:	TBD
WizardXm*Inactive Label.foreground:	TBD
To display noneditable text:	
WizardXm*DataLabel.background:	TBD
WizardXm*DataLabel.foreground:	TBD
WizardXm*DataLabel.marginHeight:	0
WizardXm*DataLabel.marginWidth:	0
List	
WizardXm*XmList.background:	TBD
WizardXm*XmList.borderWidth:	0
WizardXm*XmList.foreground:	TBD
WizardXm*XmList.marginHeight:	0
WizardXm*XmList.marginWidth:	0

Table A-3. Resources and settings for the Wizard toolkit (continued).

Resource	Setting
Menubutton Push Button	
The push button in a window that users press to display a popup list:	
WizardXm*menubutton.armColor:	TBD
WizardXm*menubutton.background:	TBD
WizardXm*menubutton.foreground:	TBD
The popup list that is displayed as a result of selecting the push button:	
WizardXm*menubuttonlist.background:	TBD
WizardXm*menubuttonlist.foreground:	TBD
Option Menu	
WizardXm*option*background:	TBD
WizardXm*option*foreground:	TBD
Page Editor	
WizardXm*PageEditor.background:	TBD
WizardXm*PageEditor.foreground:	TBD
WizardXm*PageEditorRegion.background:	TBD
WizardXm*PageEditorRegion.foreground:	TBD
Note: These resources reflect work in progress; they are not yet a published Wizard API but will be available in a future software release.	
Popup Menu	
WizardXm*popup*background:	TBD
WizardXm*popup.fillOnArm:	False
WizardXm*popup*foreground:	TBD
Pulldown Menu	
WizardXm*pulldown.adjustLast:	False
WizardXm*pulldown*background:	TBD
WizardXm*pulldown*foreground:	TBD
PushButton	
WizardXm*pushbuttons.armColor:	TBD
WizardXm*pushbuttons.background:	TBD
WizardXm*pushbuttons.foreground:	TBD
WizardXm*CANCELbuttons.background:	TBD
WizardXm*CANCELbuttons.bottomShadowColor:	TBD
WizardXm*CANCELbuttons.foreground:	TBD

WizardXm*CANCELbuttons.armColor:	TBD

Table A-3. Resources and settings for the Wizard toolkit (continued).

Resource	Setting
WizardXm*EXITbuttons.background:	TBD
WizardXm*EXITbuttons.bottomShadowColor:	TBD
WizardXm*EXITbuttons.foreground:	TBD
WizardXm*EXITbuttons.armColor:	TBD
WizardXm*OKbuttons.background:	TBD
WizardXm*OKbuttons.foreground:	TBD
WizardXm*QUITbuttons.background:	TBD
WizardXm*QUITbuttons.bottomShadowColor:	TBD
WizardXm*QUITbuttons.foreground:	TBD
WizardXm*QUITbuttons.armColor:	TBD
ScrollBar	
WizardXm*XmScrollBar.troughColor:	TBD
WizardXm*XmScrollBar.background:	TBD
Separator	
WizardXm*XmSeparator.background:	TBD
Text Field (Editable)	
WizardXm*XmTextField.background:	TBD
WizardXm*XmTextField.borderWidth:	0
WizardXm*XmTextField.foreground:	TBD
WizardXm*XmTextField.highlightThickness:	2
WizardXm*XmTextField.highlightOnEnter:	False
WizardXm*XmTextField.marginHeight:	0
WizardXm*XmTextField.marginWidth:	2
WizardXm*XmTextField.shadowThickness:	0

Note: On a Sun keyboard, the F20 key should be used:

```
WizardXm*XmText.translations:  #override\n\
                    <Key>F20:  beginning-of-line()  delete-to-end-of-line() \n\
                    s ~c ~m ~a <Key>space:  replace_action() \n\
                    <Key>:  replace_action() /n/
```

Note: On a Hewlett-Packard keyboard, the QuickPaste key should be used:

```
WizardXm*XmText.translations:  #override\n\
                    <Key>osfQuickPaste:  beginning-of-line()  delete-to-end-of-line() \n\
                    s ~c ~m ~a <Key>space:  replace_action() \n\
                    <Key>:  replace_action() /n/
```

ToggleButton

Resource	Setting
WizardXm*XmToggleButton.background:	TBD

WizardXm*XmToggleButton.bottomShadowColor: TBD

Table A-3. Resources and settings for the Wizard toolkit (continued).

Resource	Setting
WizardXm*XmToggleButton.foreground:	TBD
WizardXm*XmToggleButton.selectColor:	TBD
WizardXm*XmToggleButton.topShadowColor:	TBD

APPENDIX B
USER INTERFACE SPECIFICATIONS CHECKLIST

The checklist provided in this appendix was developed as a reference aid for developers as they implement the specifications in their applications. Developers should use the checklist as a supplement to, rather than substitute for, the detailed provisions presented in this document. Developers are encouraged to use the checklist to evaluate the user interface for their applications. Provisions in the checklist labeled as mandatory are required so that applications comply with the Motif Style Guide. Provisions labeled as recommended are based on published user interface guidelines and are included to ensure consistency between applications. Developers shall comply with the mandatory provisions in the checklist unless an exception is needed to provide critical functionality within the application. Developers should comply with the recommended provisions that are applicable to and appropriate for the functions available within the application.

The checklist contains one-sentence, single-line statements of the provisions included in sections 2 through 10 of the specifications. Items that are mandatory provisions are labeled “M;” items that are recommended provisions are labeled “R.” The <> used to designate virtual keys is omitted in order to conserve space. The checklist follows the overall organization used in the document, with the items in each section of the checklist presented in the same order that they are discussed in the document.

The checklist is updated to reflect changes to the document as new versions are published. An electronic version of the checklist can be requested by sending e-mail to fernande@nosc.mil.

USER INTERFACE SPECIFICATIONS CHECKLIST

INTERCHANGEABILITY BETWEEN INPUT DEVICES

_____ M 1 The pointing device is the primary means of user-computer interaction.
_____ M 2 The keyboard is available for performing most operations primarily as a back-up.
_____ M 3 All operations in table 2-2 except range selection in text are available from keyboard.
_____ R 4 Range selection in text is included if integral to the functionality of an application.

POINTING DEVICE INPUT

The Pointer

_____ M 1 The pointing device is associated with a single pointer on the screen.
_____ M 2 The hotspot of the pointer indicates the precise location where operations occur.
_____ M 3 The pointer moves anywhere on the screen.
_____ M 4 When users move the pointing device, the pointer moves in the corresponding direction.
_____ R 5 The pointing device-to-pointer movement ratio is close to 1:1 for most user interactions.
_____ R 6 Users can select or adjust the rate sensitivity of the pointing device.
_____ R 7 The pointer remains in place until moved by users; it is not moved by an application.
_____ M 8 The pointer moves between multiple screens when users move the pointing device.
_____ M 9 The pointer does not move beyond the screen boundaries or disappear from sight.
_____ M10 The location of the hotspot does not move as the pointer changes shape.

Pointer Shapes

_____ M 1 The pointer shapes in table 2-1 are used when providing the functions indicated.
_____ M 2 An applications redefines pointer shape only when pointer is in application window.
_____ M 3 The upper-left-pointing arrow is used for object selection in most windows.
_____ M 4 The X pointer shape is not used by an application.
_____ R 5 New pointer shapes are not created for functions that already have a shape.

_____ R 6 Pointer shapes are not associated with functions they were not designed to represent.
_____ R 7 New pointer shapes are easy to see, with a hotspot that is obvious and easy to locate.
_____ R 8 New pointer shapes suggest their purpose and are not confused with other objects.

Pointing Device Buttons

_____ M 1 The Select (S) function is bound to the left button on two- or three-button pointing device.
_____ M 2 The Transfer (T) function is bound to the middle button on three-button pointing device.
_____ M 3 The Menu (M) function is bound to the right button on three-button pointing device.
_____ R 4 Application functions bound to the T button do not conflict with other system functions.
_____ R 5 The Transfer function is bound to the right button on a two-button pointing device.
_____ R 6 The Menu function is bound to the two buttons together on a two-button pointing device.
_____ R 7 Left-handed users can exchange the functions between the left and right buttons.
_____ R 8 The double-click speed can be easily executed by users and can be modified if desired.
_____ R 9 The pointing device may provide a velocity sensitive gain adjustment.

KEYBOARD INPUT

The Location Cursor

_____ M 1 The object that has keyboard focus in a window is identified by a location cursor.
_____ M 2 Location cursor is controlled from keyboard, is not affected by pointing device movement.
_____ M 3 When users click on an object, it receives focus and the location cursor moves to the object.
_____ M 4 When a window is displayed, location cursor is on the control most likely to be selected.
_____ M 5 When a window regains focus, the location cursor is on the control that last had focus.
_____ M 6 The box cursor is the default shape for the location cursor.
_____ M 7 Text cursor is displayed in text with keyboard focus, indicates where typed text appears.
_____ M 8 The text cursor shape is a vertical bar in both insert and replace modes.

_____ M 9 The flash rate for the text cursor is 2-5 Hz.
_____ M10 When the text with the cursor loses focus, the cursor is grayed out and stops flashing.
_____ M11 When text regains focus, text cursor returns to normal appearance and resumes flashing.
_____ M12 If text cursor disappears when focus lost, it reappears at same place when focus returns.
_____ M13 Only one location cursor appears in a window at any time.
_____ M14 Only one text cursor appears in a window at any time.
_____ M15 Placing the location cursor on a text object also places the text cursor in that object.
_____ R 16 Existing cursors are used; new ones created if existing do not provide functions desired.

Key Assignments

_____ M 1 All operations in table 2-2 except range selection in text are available from keyboard.
_____ M 2 The keyboard mappings in table 2-3 are used to perform the operations in table 2-2.
_____ R 3 New key bindings are consistent with those used by other applications in the system.
_____ R 4 The combination of Alt and alphanumeric characters is used only in mnemonics.
_____ R 5 The use of Shift, Alt, and Ctrl with other keys is limited to keyboard accelerators.
_____ R 6 New bindings are “visible” in a window (e.g., as mnemonics/keyboard accelerators).
_____ R 7 Command names for variable function keys can be displayed on the screen as soft keys.
_____ R 8 Soft key labels displayed on the screen are modified when the meaning of a key changes.
_____ R 9 No more than two functions are assigned per soft key.
_____ R 10 Users can provided with an easy means to return to the set of base-level functions.
_____ R 11 The actions mapped to soft keys do not conflict with those included in this style guide.
_____ R 12 The actions mapped to soft keys do not conflict with those used by other applications.

Text Entry

_____ R 1 When the pointer moves into a text area, it changes to an I-beam shape.
_____ M 2 When users click the S button in text area, text cursor appears at the pointer location.

_____ M 3 If a text area is empty, the text cursor shall appear at the beginning of the area.
_____ M 4 If a text area has text, the text cursor appears between the characters under the pointer.
_____ M 5 If the pointer is beyond the end of text, the text cursor follows the final text character.
_____ M 6 When a window opens, text cursor appears in text area where typing is likely to occur.
_____ M 7 When text cursor moves between fields, it appears at beginning of text in the field.
_____ M 8 Return does not move text cursor between fields; it is used only for default push button.
_____ R 9 The text cursor appears only in text entry areas and not where text entry is not possible.
_____ R 10 Text entry is possible only after the text cursor is visible at a legal location.
_____ R 11 Text entry is not possible when the text cursor is not visible.
_____ R 12 The text cursor is highly visible whenever it appears in a text entry area.

Text Entry Modes

_____ M 1 Insert mode adds new text to the left of the text cursor and moves existing text to right.
_____ M 2 Replace mode replaces character to right of text cursor with new character typed.
_____ M 3 Backspace deletes the character to the left of the text cursor.
_____ M 4 Delete deletes the character to the right of the text cursor.
_____ R 5 Double clicking on text selects and highlights the word at the location of the pointer.
_____ R 6 Delete deletes highlighted text; text cursor appears; other text is compressed.
_____ M 7 Insert toggles between text entry modes.
_____ M 8 The text cursor does not change shape when users change between text entry modes.
_____ M 9 Whenever the text cursor moves into a text field, the default is insert mode.
_____ R 10 Applications provide users with access to both text entry modes.
_____ R 11 Applications do not switch arbitrarily between modes as users move between text fields.

ALTERNATE INPUT DEVICES

_____ R 1 If alternate input devices are used, interaction is consistent with models presented here.

WINDOW NAVIGATION

Input Focus Policy

_____ M 1 Only one window on the screen has input focus at any time.
_____ R 2 Explicit focus is used with overlapping placement, implicit focus with tiled placement.
_____ M 3 The default focus policy is explicit but users can switch to implicit if desired.
_____ M 4 Users assign focus with either with pointing device or from keyboard.
_____ R 5 The focus policy selected by users is applied to all applications within a system.
_____ R 6 Applications are designed to support both explicit and implicit focus.
_____ M 7 When focus is implicit, keyboard focus moves with the pointer.
_____ R 8 The location cursor is displayed regardless of the focus policy in effect.

Assigning Focus to a Window

_____ M 1 Users assign focus by moving the pointer into a window and clicking the S button.
_____ M 2 If users click in an empty window area, the frame highlights.
_____ M 3 If users click in the title bar, the frame highlights and the window is raised to the front.
_____ M 4 If users click on object, the frame highlights, window is raised, and object is selected.
_____ M 5 A parent window automatically regains focus when a child window is closed.
_____ M 6 Alt+Tab and Alt+Shift+Tab move focus forward & backward through window families.
_____ M 7 Alt+F6 and Alt+Shift+F6 move focus forward & backward through windows in family.

NAVIGATION WITHIN WINDOWS

_____ M 1 Only one object in the window with input focus receives input from the keyboard.
_____ M 2 The location cursor indicates the object with keyboard focus.
_____ M 3 When a window receives focus, the location cursor appears on object that last had focus.
_____ M 4 Users perform navigation and selection with either pointing device or keyboard.

Pointing Device Navigation for Controls

_____ M 1 Placing pointer on object & clicking S button moves location cursor to object & gives focus.

_____ M 2 The shape of the location cursor (usually a box) is appropriate to the type of object.
_____ M 3 Ctrl and clicking S button on an object moves location cursor to object but does not select it.
_____ R 4 Autoscrolling is available when pointer is on a scrollable control such as text or a list.

Keyboard Navigation for Controls

_____ R 1 Keys used to navigate between/within tab groups fit task and support performance.
_____ M 2 Ctrl+Tab and Ctrl+Shift+Tab move location cursor to the next and previous tab group.
_____ M 3 Tab and Shift+Tab also move to the next and previous tab group except in multi-line text.
_____ M 4 Location cursor is on default or first control when location cursor moves to a tab group.
_____ M 5 Direction of cursor movement is from upper left to lower right, with wrapping.
_____ M 6 Location cursor skips a tab group if none of the controls can have keyboard focus.
_____ M 7 Up, Down, Left, and Right move location cursor between controls in tab group with focus.
_____ M 8 Moving the location cursor to a control does not change the state of the control.
_____ M 9 Cursor movement is upper left to lower right, with wrapping, unless control is scrollable.
_____ M10 Arrows move location cursor one increment; Ctrl+arrow keys one large increment.
_____ M11 Home and End move to leftmost/rightmost element in a control.
_____ M12 Ctrl+Home and Ctrl+End move to beginning/end element in a control.
_____ M13 PageUp, PageDown, PageLeft, and PageRight scroll one page (minus one line).
_____ M14 Focus remains on element where it was before scrolling began and may not be in view.
_____ M15 When keyboard action alters element with focus, scrolling occurs so element is in view.

Location Cursor Behavior During Keyboard Navigation

_____ R 1 The tab groups in a window support efficient navigation among sets of related controls.
_____ R 2 The sequence of tab groups matches the order users will interact with the controls.
_____ R 3 Scroll bars are included as tab groups within a window.

_____ R 4 Default position for location cursor is the control with which users will interact first.
_____ R 5 When a window opens, location cursor appears in its default position in the window.
_____ M 6 One active location cursor is displayed in a window at a time.
_____ R 7 The location cursor is always visible as it moves between/within tab groups in a window.
_____ R 8 Position of the location cursor is not affected by movement of the pointer in a window.
_____ R 9 When users move the location cursor to a control, the state of the control does not change.
_____ R 10 Users make a selection while location cursor is on a control to change the state of control.

OBJECT SELECTION

_____ R 1 Method used in a collection allows selections to be made quickly and without error.
_____ R 2 Same selection method is used whenever the same type of collection is presented.
_____ R 3 Selection method(s) for an object matches the type of action executed on the object.
_____ M 4 When an element is selected, its appearance changes to indicate the change in its state.

Pointing Device Selection Methods

_____ M 1 Click the S button on an element to select it; other selected elements are deselected.
_____ M 2 Click S button on multiple elements one at a time to select them; highlight as selected.
_____ M 3 Drag the pointer over contiguous elements in a range with the S button to select them.
_____ M 4 Extend range selection by pressing Shift & clicking S button on last element in the range.
_____ M 5 Add/Remove a discontiguous element by pressing Ctrl & clicking S button on element.
_____ M 6 Add/Remove discontiguous range by pressing Ctrl & dragging with S button over range.
_____ M 7 A bounding box appears when dragging the pointer over elements in a 2-D collection.
_____ R 8 Users can quickly select and deselect all elements in a collection with pointing device.
_____ R 9 Users can quickly select an element from a group of closely overlapping elements.

Keyboard Selection Methods

_____ M 1 The location cursor is a solid rectangle in normal mode, a dotted rectangle in add mode.
_____ M 2 Add mode is used to select one element or multiple elements one at a time.
_____ M 3 Normal mode is used to select multiple contiguous elements; add mode may also be used.
_____ M 4 Add mode and normal mode are used to select multiple discontiguous elements.
_____ M 5 Space (and Select) selects an element or multiple elements one at a time.
_____ M 6 Space (and Select) sets the anchor for selecting a range of contiguous elements.
_____ M 7 Shift+Space (and Shift+Select) extends selection from anchor to last element selected.
_____ M 8 Shift+F8 toggles between add mode and normal mode.
_____ M 9 Ctrl+/ selects all of the elements in a collection.
_____ M10 Ctrl+\ deselects all of the elements in a collection.
_____ M11 Cancel undoes a selection action and returns the elements to their normal appearance.

Other Types of Selection

_____ M 1 A default action in a window is executed by double clicking when making a selection.
_____ M 2 Enter or Ctrl+Return invokes the default action after making a selection in a window.
_____ M 3 Return invokes default action in a window if focus is on object other than multi-line text.
_____ R 4 Expert activation is used only as short-cut to features available elsewhere in a window.
_____ R 5 If expert activation is available, users can reverse the effects of the action.
_____ R 6 The default action is the same for every object in a given category (e.g., text, lists).
_____ R 7 The default action is the action that users are most likely to execute on the object.
_____ R 8 Extended selection is available in text.
_____ R 9 Click places the text cursor, double click selects a word, triple click selects a line.
_____ M10 Cancel cancels action being executed and returns object to its state prior to action.

OBJECT TRANSFER

_____ R 1 Objects can be transferred in a window or to another window in same/other application.

Drag Transfer

_____ M 1 Drag move: Hold down Shift while dragging the object using the T button.
_____ M 2 Drag copy: Hold down Ctrl while dragging the object using the T button.
_____ R 3 M: Drag link: Hold down Ctrl+Shift while dragging the object using the T button.
_____ M 4 Default is to use the T button to do a drag transfer; result is a move.
_____ M 5 Cancel cancels a drag operation and returns to object being dragged to original location.
_____ M 6 Elements moved within a component remain selected after they have been moved.
_____ M 7 Dragging a set of selected elements usually drags the entire collection.
_____ M 8 Dragging an unselected element in lists and graphics affects only the element.
_____ M 9 Dragging in overlapping elements occurs on the highest draggable element in the stack.
_____ M10 Pointer shape changes to a drag icon during the drag operation and then back to pointer.
_____ M11 The drag icon contains a source indicator and may contain operations & state indicators.
_____ R12 New drag icons are easy to see, with a hotspot that is obvious and easy to locate.

Clipboard Transfer

_____ M 1 Cut clears clipboard, stores a copy of the object in clipboard, removes object from window.
_____ M 2 If cut object is graphic, previous space is left blank; if text, remaining text is compressed.
_____ M 3 Copy clears clipboard & stores object copy in clipboard; object stays in original location.
_____ M 4 Paste copies the object in the clipboard to a new location.
_____ M 5 If object is text, paste copies object to location of text cursor; text appears to left of cursor.
_____ M 6 If object is graphic, paste copies to pointer location in window with input focus.
_____ M 7 Pasted object remains in the clipboard until another object is cut/copied into it.
_____ R 8 Pasting an object from the clipboard does not select the object.
_____ M 9 Clipboard link places link in clipboard to selected objects, transfers link to new location.

_____ R 10 A link is placed by executing a Paste or Paste Link action.
_____ R 11 Clipboard transfer is available whenever an editable object has keyboard focus.
_____ M12 Cut (and Shift+Delete) performs a cut operation.
_____ M13 Copy (and Ctrl+Insert) performs a copy operation.
_____ M14 Paste (and Shift+Insert) performs a paste operation.
_____ R 15 Accelerators are available to perform other editing operations (e.g., Clear, Delete).
_____ R 16 Access to clipboard transfer is provided in consistent fashion throughout an application.
_____ R 17 Access to clipboard transfer is provided when appropriate to the task being performed.
_____ R 18 Users can view clipboard contents, are informed when cut / copy object of excessive size.

Primary Transfer

_____ M 1 Primary move: Select object, place pointer at destination, press Shift & click T button.
_____ M 2 Primary copy: Select object, place pointer at destination, press Ctrl & click T button.
_____ M 3 Primary link: Select object, place pointer at destination, press Ctrl+Shift & click T btn.
_____ M 4 Transferring an object by primary copy or link does not select the object.
_____ M 5 Transferring an object via a primary move selects the object.
_____ M 6 When the T button is used to perform a primary transfer, the default is a copy operation.
_____ M 7 Alt+Copy (and Alt+Ctrl+Insert) performs a primary copy.
_____ M 8 Alt+Cut (and Alt+Shift+Delete) performs a primary move.

Quick Transfer

_____ M 1 Quick move: Hold down Alt+Shift while dragging the object using the T button.
_____ M 2 Quick copy: Hold down Alt+Ctrl while dragging the object using the T button.
_____ M 3 Quick link: Hold down Alt+Ctrl+Shift while dragging the object using the T button.
_____ M 4 When Alt and the T button are used for quick transfer, the default is a copy operation.
_____ M 5 Using quick transfer does not select the object being transferred.

INTERACTIVE CONTROL

Object-Action Selection

_____ R 1 Users first select an object, then select an action to perform on that object.
_____ R 2 Users are informed when interaction diverges from object-action selection paradigm.

User Control of Interaction

_____ R 1 Users control pace of interaction with an application, not forced to specific rate.
_____ R 2 The application executes an action only in response to explicit user input.
_____ R 3 The pace of user input does not slow down the speed of application processing.

Immediate Feedback

_____ R 1 When users take an action, there is an immediate and visible response to the action.
_____ R 2 A visible response occurs even if the result cannot be displayed immediately.
_____ R 3 Visual cues show application accepting input / temporarily unavailable / unavailable.
_____ R 4 The appearance of an object provides an indication of its availability.
_____ R 5 If an operation requires several actions, users are prompted with the actions to take.
_____ R 6 Applications ignore user actions made during periods when input cannot be accepted.
_____ R 7 The pointing device and / or keyboard are disabled when input may be destructive.
_____ R 8 Users cannot override disabling but are able to stop a process.

Response Time

_____ R 1 System response is within .2 sec of user action; display takes no more than .5-1.0 sec.
_____ R 2 Requests for new displays can take 2-10 sec. if operation requires extensive processing.
_____ R 3 Error feedback is provided to users within 2 sec. of the time error was detected.
_____ R 4 When a user request takes more than 2 sec. to process, pointer shape changes to a watch.
_____ R 5 When user request takes more than 5 sec., message window informs operation is lengthy.

Error Detection

_____ R 1 Application does not execute an invalid action, only displays message.
_____ R 2 When users make multiple errors with a single action, they are notified of each error.
_____ R 3 Feedback is immediate, is visual and/or auditory, and explains the nature of the error.
_____ R 4 When an error is repeated, feedback shows that attempted correction was processed.
_____ R 5 Users are required to correct only the invalid action and not to repeat the entire sequence.
_____ R 6 After making correction, users execute same action for re-entry that was used originally.

Explicit Destruction

_____ M 1 Users confirm destructive action before action is executed by the application.
_____ R 2 If destructive action applies to multiple objects, list is shown to select ones that apply.
_____ R 3 Users confirm a close-window action only when the action will cause loss of data.
_____ R 4 Window remains displayed while request to confirm destructive action is presented.

General "Undo" Capability

_____ R 1 Users can "undo" the most recent selection/action except for explicit destruction.
_____ R 2 Undo can deselect objects, return to a prior state, & retrieve previous screen information.
_____ R 3 Irreversible actions are labeled and clearly separated from those that are not.

Use of Processing Modes

_____ R 1 Processing modes are not used; the same action has the same effect whenever executed.
_____ R 2 If a mode is used, a visual cue is provided to indicate the mode currently in effect.
_____ R 3 The application minimizes extent to which user interaction is limited by Motif modes.

CONSISTENCY IN PERFORMING OPERATIONS

_____ R 1 Sequences of actions taken by users allow completion rapidly and with minimum steps.

_____ R 2 Each operation is done by following same action sequence consistently in an application.
_____ R 3 Other approaches to operation can be provided, but standard one is always available.
_____ R 4 Users execute only those actions that are required to perform an operation.
_____ R 5 Users perform an operation the same way in different applications within a system.
_____ R 6 The frequency with which users move between input devices in a task is minimized.

WINDOW COMPONENTS

The Window Menu

_____ M 1 Window Menu in primary window contains all standard window functions.
_____ M 2 Window Menu in secondary window contains options for the components included.
_____ M 3 Lower in a secondary window lowers all of the children of the window's parent.
_____ M 4 Window Menu in a secondary window includes Move, Lower, & Close but not Minimize.
_____ M 5 Spring-loaded & posted methods are used to display Window Menu and select options.
_____ M 6 Double clicking the Window Menu button in a window closes the window.
_____ M 7 Shift+Escape or Alt+Space displays the Window Menu.
_____ M 8 Navigation and selection is done as in a pull-down menu.

Components of Primary and Secondary Windows

_____ M 1 A primary window contains all of the standard window components.
_____ M 2 A secondary window contains a title area and a Window Menu button.
_____ M 3 A secondary window contains resize borders & Maximize button if window can be resized.
_____ R 4 A secondary window contains a Minimize button if window can be minimized.
_____ R 5 Each window contains only components that are appropriate to that window type.
_____ R 6 The components on a window behave in a consistent manner.
_____ R 7 Each window contains only one set of window components that are selectable by users.
_____ R 8 If needed, additional window functions are mapped to buttons next to existing ones.

_____ R 9 Users can move a dialog window but not minimize, maximize, or resize it unless required.

WINDOW MANAGEMENT FUNCTIONS

Pointing Device Interaction

_____ R 1 When window is closed & then reopened, it is displayed where it was when closed.
_____ M 2 Move: Select Move option in Window Menu, then drag the window to new location.
_____ M 3 Move: Press the S or T button on the title bar, then drag window to new location.
_____ M 4 Move: An outline of the window moves on the screen as users move the pointer.
_____ M 5 Resize: Select the Size option in Window Menu, then drag frame to new size.
_____ M 6 Resize: Press the S or T button on the window frame, then drag it to new size.
_____ M 7 Resize: An outline of the window moves on the screen as users move the pointer.
_____ M 8 Minimize: Select the Minimize option in the Window Menu.
_____ M 9 Minimize: Select the Minimize button on the title bar.
_____ M10 Minimize: Window is minimized, and icon appears in the lower left corner of screen.
_____ M11 Restore: Double click the S button on an icon to restore a minimized window.
_____ M12 Maximize: Select the Maximize option in the Window Menu.
_____ M13 Maximize: Select the Maximize button on the title bar.
_____ M14 Maximize: The window expands to its full size.
_____ M15 Raise: Click on the title bar to raise window to front of screen and give it input focus.
_____ M16 Lower: Select Lower option in Window Menu to move window to bottom of hierarchy.
_____ M17 Close: Select the Close option in the Window Menu.
_____ M18 Close: Double click on the Window Menu button.
_____ M19 Close: The window is closed and users are prompted to save any unsaved data.
_____ M20 Close: When window is closed, focus is not arbitrarily assigned to another window.

TYPES OF WINDOWS

Primary Windows

_____ M 1 Minimizing a primary window replaces it and all its secondary windows with an icon.
_____ M 2 When a primary window is minimized, processing in the window continues.
_____ M 3 Opening an icon redisplays the primary window and its secondary windows.
_____ M 4 Each primary window in an application can be minimized separately.
_____ M 5 Closing a primary window removes it and its secondary windows from the screen.
_____ M 6 When a primary window is closed, processing in it stops.
_____ M 7 When the last primary window for an application is closed, the application is closed.

Secondary Windows

_____ M 1 When secondary window is opened, it appears in front of parent which stays displayed.
_____ M 2 When a secondary window is closed, its children are closed but its parent is not affected.
_____ R 3 Dialog windows have the lowest level of modality so users are interrupted minimally.
_____ R 4 Menu windows are modeless.

Window Placement

_____ R 1 Window management functions in overlapping placement are available to users.
_____ R 2 Tiled placement restricts window management functions (no move, resize, close).

_____ R 3 Overlapping placement is system default; tiled placement available if required.
_____ R 4 Users can arrange windows, save and retrieve configuration as a user-preference setting.

WINDOW ICONS

Appearance

_____ M 1 An icon has the same title as its corresponding window.
_____ M 2 An icon title is the same width as the icon image; may be truncated to fit.
_____ M 3 The location cursor appears on the icon with input focus, and full icon title is displayed.
_____ M 4 When an icon loses input focus, the title is truncated to same width as the icon image.

_____ M 5 The default location for icons is the lower left corner of the screen.
_____ M 6 If more than one window is minimized, the icons are arrayed in rows from left to right.
_____ R 7 An option to change the default icon location is available as a user preference setting.

Design of Window Icons

_____ R 1 Icons have unique graphic images so that users can recognize the application/function.
_____ R 2 Default size of the icon image is 50 x 50 pixels; maximum size is 64 x 64 pixels.
_____ R 3 Icons use a concrete, rather than abstract, image to represent the application/function.
_____ R 4 Icons are visually distinct from one another so users can easily discriminate among them.
_____ R 5 The icons for the objects within a class have the same shape but vary in their details.
_____ R 6 The graphic in an icon provides a simplified representation of the object.
_____ R 7 Variation in angles, line thicknesses, shapes, empty space is limited.
_____ R 8 Icon graphics in application are consistent with icons in other applications in system.
_____ R 9 Graphic image is monochromatic, in color similar to other text/graphics in system.
_____ R 10 Use of text as part of the icon graphic is minimized.
_____ R 11 Icon changes to indicate when routine message generated by process running in window.

Behavior

_____ M 1 Icon Menu has same options (except Size) as Window Menu for corresponding window.
_____ M 2 Minimize is included in an Icon Menu but is not available for selection.
_____ M 3 Users click the S button on the icon to display the Icon Menu.
_____ M 4 Users dismiss the Icon Menu by clicking the S button anywhere outside the menu.

PULL-DOWN MENUS

Menu Title

_____ R 1 The title of a menu is a single word whenever possible.
_____ M 2 The title of a pull-down menu is displayed in a menu bar at the top of a window.

_____ R 3 A menu title describes the category or type of options and is different from other titles.
_____ R 4 The first letter of each word in the menu title is capitalized.
_____ R 5 If the title contains an acronym, it is capitalized.
_____ R 6 The title does not contain an ellipsis or a right-pointing arrow.

Types of Menu Options

_____ M 1 A routing option that displays a window is followed by an ellipsis.
_____ M 2 A routing option that displays a cascading submenu is followed by right-pointing arrow.
_____ R 3 Wording of an action toggle option reflects the action implemented when option selected.
_____ R 4 Wording of an action toggle option is semantically congruent with natural usage.
_____ R 5 When action toggle selected, its wording changes to reflect when action selected again.
_____ M 6 Only one of the actions for a toggle appears in the menu at any time.
_____ R 7 Wording of an “undo” option changes dynamically to reflect the action to be undone.
_____ R 8 Wording of state toggle describes the state and may include radio/check buttons at left.
_____ R 9 When state toggle is selected, the button highlights without the wording changing.

Wording

_____ R 1 Menu options are phrased to reflect the action executed and worded in user vocabulary.
_____ R 2 The vocabulary in table 6-1 is used when the actions in table are included in an option.
_____ R 3 Options are tersely worded and in upper/lower case letters; first letter is capitalized.
_____ R 4 Acronyms are capitalized.
_____ R 5 Each option is left-justified and appears on a single line.
_____ R 6 The wording of each option is consistent in grammatical style, matches the menu title.
_____ R 7 Verbs are used as the first word in a menu option.
_____ R 8 Supplementary information about menu options is available in message area of window.

Organization and Grouping

_____ R 1 The menu is wide enough for easy reading of the longest option and accelerator.

_____ R 2 Menu options are organized in logical or functional groupings, with clear titles.
_____ R 3 If not in logical groups, order is by frequency of usage, with most frequent at the top.
_____ R 4 If not in logical groups or by frequency, options are in alphabetical/numerical order.
_____ R 5 Less frequently executed options and destructive options are at the bottom of the menu.
_____ R 6 Options that perform opposing actions are not placed adjacent to each other.
_____ R 7 If similar options are in different menus, the options are ordered in a consistent manner.
_____ R 8 A menu contains no less than three options or more than ten options.
_____ R 9 Menu with more than 4 options is divided into groups of 4 options separated with line.

Availability

_____ R 1 If option or set of options is never available to a user, the option(s) is not in a menu.
_____ R 2 If option is temporarily unavailable, it is displayed in menu but dimmed.

Cascading Submenus

_____ M 1 Cascading submenus appear to right of parent menu (below if space to right is limited).
_____ R 2 Option that is parent for cascading submenu is always displayed as available.
_____ R 3 When parent option selected, submenu is displayed, even if all options unavailable.
_____ R 4 Cascading submenus are limited to three levels.
_____ R 5 A submenu contains only the options in the submenu; does not repeat the parent option.
_____ R 6 A submenu is positioned to align the first option with arrow in the parent option.

Pointing Device Navigation and Selection

_____ M 1 Spring-loaded and posted methods are used to display a menu, select an option.
_____ M 2 When menu is displayed, the location cursor is on the first available option in the menu.
_____ M 3 Spring-loaded: Options highlight, submenus displayed as pointer dragged over them.

_____ M 4 Spring-loaded: Drag location cursor to option desired and release S button to select.
_____ M 5 Spring-loaded: Move pointer off menu and release S button to not select option.
_____ M 6 Posted: Click S button on menu title and menu remains displayed.
_____ M 7 Posted: Click S button on option; location cursor moves to option and it is selected.
_____ M 8 Posted: Move pointer off menu and click S button to not select option.

Keyboard Navigation and Selection

_____ M 1 F10 (and Shift+Menu) moves location cursor to first available menu title in a menu bar.
_____ M 2 If none of the menu titles is available, these keys do not move the location cursor.
_____ M 3 Left and Right move the location cursor between available menu titles, with wrapping.
_____ M 4 F10 (and Shift+Menu) exit menu bar & return location cursor to previous object with focus.
_____ M 5 Down displays the menu for the title containing the location cursor.
_____ M 6 The arrow keys move the location cursor between available options in the menu.
_____ M 7 Right displays a cascading submenu if option with location cursor is parent for the menu.
_____ M 8 Return, Enter, or Space (and Select) selects an option and dismisses the menu.
_____ M 9 Cancel dismisses the menu without making a selection.

TEAR-OFF MENUS

Appearance

_____ M 1 A tear-off has a dashed-line graphic which is the first menu option below the title.
_____ M 2 The title of a menu window is the title of the associated pull-down menu.
_____ M 3 Options in menu window are dimmed when unavailable, with wording that may change.
_____ R 4 The contents of the menu window are the same as the original menu, in the same order.
_____ R 5 Availability of options in menu window is managed the same way as in original menu.

Behavior

_____ M 1 Users click S button on tear-off graphic in menu to display a menu window.
_____ M 2 Users drag tear-off graphic with T button to display and move a menu window.
_____ M 3 Users click S button on option in a menu window to select it; window remains displayed.
_____ M 4 Close (in Window Menu) and Cancel (on the keyboard) close a menu window.
_____ R 5 Users can select options from the menu or the menu window.
_____ M 6 When menu is re-selected, original menu window is dismissed and new window appears.
_____ M 7 With cursor on graphic, Return, Enter, or Space (and Select) displays menu window.
_____ M 8 Arrow keys move location cursor between options; same keys as above select an option.
_____ M 9 Close (in Window Menu) and Cancel (on the keyboard) close a menu window.

POP-UP MENUS

Appearance

_____ R 1 Menu contents vary with window component, pointer position when menu is displayed.
_____ R 2 Pop-up menus do not contain cascading submenus.
_____ R 3 The options in a pop-up menu include mnemonics but not keyboard accelerators.
_____ R 4 The options in a pop-up menu are dimmed when unavailable.
_____ R 5 The order of options in a pop-up menu is the same as in corresponding pull-down menu.
_____ M 6 When pointing device is used, pop-up menu contents relate to element with pointer.
_____ M 7 When keyboard is used, pop-up menu contents relate to element with location cursor.
_____ R 8 A pop-up menu appears near the element with which it is associated.
_____ R 9 A window containing a pop-up menu provides an indication that the menu is available.
_____ R10 A pop-up menu provides redundant access to actions and does not access new actions.
_____ R11 The keyboard accelerators in pop-up menus are the same as in pull-down menus.
_____ R12 A pop-up menu contains no more than 15 options.

Behavior

_____ M 1 Spring-loaded and posted methods are used with M button to display a pop-up menu.
_____ M 2 When pop-up menu is displayed, location cursor is on first available option in the menu.
_____ M 3 Spring-loaded, posted methods used with S & M buttons display/select in pop-up menu.
_____ M 4 Shift+F10 (and Menu) displays a pop-up menu.
_____ M 5 Arrow keys move location cursor between available options in a pop-up menu.
_____ M 6 Return, Enter, or Space (and Select) selects an option and dismissed a pop-up menu.
_____ M 7 When option is selected with pointing device/keyboard, the pop-up menu is dismissed.
_____ M 8 Cancel and Shift+F10 (and Menu) dismisses a pop-up menu without making a selection.

OPTION MENUS

Appearance

_____ R 1 An option menu contains no more than 15 options.
_____ R 2 The options in an option menu include mnemonics but not keyboard accelerators.
_____ R 3 The option menu button is large enough to display both longest text and bar graphic.

Behavior

_____ M 1 Spring-loaded and posted methods are used with the S button to display an option menu.
_____ M 2 When an option menu is displayed, location cursor is on the previously selected option.
_____ M 3 Spring-loaded & posted methods used with the S button navigate/select in option menu.
_____ M 4 When an option is selected in option menu, it appears as the label in option menu button.
_____ M 6 Space (and Select) displays an option menu.
_____ M 7 Arrow keys move location cursor between available options in an option menu.
_____ M 8 Return, Enter, or Space (and Select) selects option and dismisses an option menu.
_____ M 9 When an option is selected in option menu, it appears as the label in option menu button.
_____ M10 Cancel dismisses an option menu without making a selection.

MNEMONICS

Appearance

_____ R 1 Mnemonics are available for every title in a menu bar and every option in a menu.
_____ M 2 The mnemonics assigned to titles in a menu bar and to options in a menu are unique.
_____ R 3 A menu title or option has the same mnemonic whenever it appears in an application.
_____ R 4 The mnemonics listed in table 5-1 are used in menu titles and options.
_____ R 5 The same mnemonic is not used for options performing opposite or contradictory actions.
_____ R 6 Similar key(s) are used in the mnemonic and keyboard accelerator for a menu option.
_____ M 7 The character assigned as the mnemonic is underlined.
_____ R 8 Whenever possible, the mnemonic is the first letter of a menu title or option.
_____ R 9 If not first letter, the mnemonic is another character in a menu title or option.
_____ M10 If mnemonic does not appear in menu title or option, it is in parentheses after label.

Behavior

_____ M 1 Alt + mnemonic for menu title moves the location cursor to the menu bar.
_____ M 2 When menu is displayed, typing mnemonic for a menu option selects it & dismisses menu.
_____ M 3 Mnemonics are not case-sensitive.
_____ M 4 If the location cursor is in the menu bar, typing a mnemonic displays the associated menu.
_____ M 5 When a menu is displayed, typing a mnemonic selects the associated option.

KEYBOARD ACCELERATORS

Appearance

_____ R 1 Accelerators are available for frequently executed menu options.
_____ R 2 Accelerators are right justified and on same line but separate from the option label.

_____ R 3 Keyboard accelerators include a plus sign to indicate keys to be pressed at the same time.
_____ R 4 Keyboard accelerators are identified by their engravings on the keyboard.
_____ R 5 Keyboard accelerators are of the form “modifier + character.
_____ R 6 The accelerator for an option assigned a mnemonic is “modifier + mnemonic.
_____ R 7 Accelerators listed in table 5-1 are used for the actions listed.
_____ R 8 The same key combinations are used for accelerators throughout application/system.
_____ R 9 Key combinations for accelerators do not conflict with mnemonics/text entry keystrokes.
_____ R10 The keyboard accelerators listed in menu options match keyboard being used.
_____ R11 All accelerators for a single action are supported even if only one is listed in the option.
_____ R12 The key mappings on each keyboard are included in the system-level Help menu.

Behavior

_____ M 1 Users can execute keyboard accelerators available in the window with input focus.
_____ R 2 Keyboard accelerators containing an alphabetic character are not case-sensitive.
_____ M 3 When accelerator typed, associated menu is displayed briefly before option is executed.

PUSH BUTTONS

Appearance

_____ R 1 Button label is in mixed case, with the first letter of each word capitalized.
_____ R 2 Push button labels are large enough to be easily read at normal viewing distances.
_____ R 3 Space between label and push button rectangle is enough to not restrict legibility.
_____ R 4 Label includes ellipsis if selecting the push button results in dialog window displayed.
_____ R 5 Button labels do not change dynamically, based on available actions within window.
_____ R 6 The push buttons in a window are the same size.
_____ R 7 Push buttons are wide enough to display the longest button label or largest action icon.

Vocabulary

_____ R 1 Push button labels are short and unambiguous.
_____ R 2 Action buttons labels describe the action taken by the application when button selected.
_____ R 3 Each push button represents a unique action that users can perform in a window.
_____ R 4 The same term for an action is used whenever users need to perform the action.
_____ R 5 A window does not contain multiple push buttons that perform the same action.
_____ R 6 When "All" is used in a push button label, there is no ambiguity as to referent.
_____ R 7 Push button labels with multiple referents include the name of the object / element.
_____ R 8 Push buttons are located near the object(s) they affect.
_____ R 9 The vocabulary in table 6-1 is used in push button labels for the actions listed.
_____ R 10 When new vocabulary is used, it describes actions not listed in the table.
_____ R 11 Push button actions are verbs, stated in active voice, and describe what the button does.
_____ R 12 The names of actions are congruent (e.g., save / delete, on / off, in / out).
_____ R 13 The same vocabulary describes the same action throughout the application.

Default Push Buttons

_____ R 1 A default push button is available in each dialog window in an application.
_____ R 2 The same push button is the default whenever a dialog window is displayed.
_____ R 3 The default push button may vary depending on the object with focus in a window.
_____ R 4 Default designation moves with location cursor in keyboard navigation in push buttons.
_____ R 5 Default designation returns to original button when focus leaves the push button group.
_____ R 6 Cancel (not OK / Yes) is default if the action executed by the button may be destructive.
_____ R 7 Whenever possible, default action is reversible.

Design of Action Icons

_____ R 1 Action icons have unique graphic images so that users recognize the action performed.
_____ R 2 Graphic image is unambiguous and easily distinguished from other action icons.

_____ R 3 Colors used in images are similar to other system colors and are used in consistent manner.
_____ R 4 Icon graphics in application are consistent with icons in other applications in system.
_____ R 5 Graphics for action icons representing opposite actions are designed to mirror each other.
_____ R 6 Graphics are presented in a common style and oriented consistently within the button.
_____ R 7 Action icons do not contain an ellipsis, even if selecting the icon displays a window.
_____ R 8 Icons depict before/after representation, the tool to perform the action, the action itself.

Behavior

_____ M 1 The S button on the pointing device is used to select a push button.
_____ M 2 Space (and Select) selects a push button from the keyboard.
_____ M 3 When a push button is selected, it highlights and the action it represents is executed.

RADIO BUTTONS

Appearance

_____ M 1 Radio buttons are used in groups to select one from multiple mutually exclusive options.
_____ R 2 The label of a radio button defines the state being set by the user.
_____ R 3 The text is displayed in mixed case, with the first letter of the option capitalized.
_____ R 4 If a radio button cannot be selected, its label is grayed out to indicate its unavailability.
_____ R 5 Radio buttons are the same size whenever they appear in a window.
_____ R 6 None is provided as option in group of radio buttons as appropriate.
_____ R 7 Users cannot deselect all of the radio buttons in a group.
_____ R 8 A group of radio buttons provides no more than eight alternatives, includes a title.
_____ R 9 The orientation for a group of radio buttons is vertical and left-aligned.
_____ R10 If placed horizontally, space is sufficient so button is paired with label on right, not left.

Behavior

_____ M 1 The S button on the pointing device is used to select a radio button.
_____ M 2 Space (and Select) selects a radio button from the keyboard.

_____ M 3 When a radio button is selected, it highlights and any other selected one is deselected.
_____ M 4 If a radio button is in a window with a default action, Enter/Return executes the action.
_____ R 5 When radio button selected, only select state changes; no action taken/window opened.

CHECK BUTTONS

Appearance

_____ M 1 A check button is a nonexclusive setting; selecting one toggles a setting or state.
_____ R 2 A check button (not two radio buttons) is used if an option can only be set to on or off.
_____ R 3 Check buttons are used singly or in related groups.
_____ R 4 Check buttons are the same size whenever they appear in a window.
_____ R 5 The number of check buttons in a group is limited to eight or less, include a title.
_____ R 6 The orientation for check buttons is vertical and left-aligned.
_____ R 7 If placed horizontally, space is sufficient so button is paired with label on right, not left.
_____ R 8 The text is displayed in mixed case, with the first letter of the option capitalized.
_____ R 9 If a check button cannot be selected, its label is grayed out to indicate its unavailability.
_____ R 10 Check buttons in rectangle form are used only in icon bars.

Behavior

_____ M 1 The S button on the pointing device is used to select a check button.
_____ M 2 Space (and Select) selects a check button from the keyboard.
_____ M 3 When a check button is selected, it highlights and any other selected on stays selected.
_____ M 4 If a check button is in a window with a default action, Enter/Return executes the action.
_____ R 5 When check button selected, only select state changes; no action taken/window opened.
_____ R 6 Selecting one check button in a group does not affect the state of any other button in group.

TEXT FIELDS

Editable Text Fields

_____ M 1 The title appears to the left or above the text field and describes what is to be entered.
_____ R 2 The title of a text field is followed by a colon.
_____ R 3 The title of a text field includes cues regarding format.
_____ R 4 If a unit of measurement is always used, it is part of title and does not have to be entered.
_____ R 5 The title is designed so that users can easily distinguish it from the text field.
_____ R 6 The title of a text field differ from the titles of other text fields.
_____ R 7 Text fields indicate whether text entry is mandatory or optional.
_____ R 8 The text field indicates the basic features of the entry required.
_____ R 9 If information being entered is a fixed length, the text field is the same length.
_____ R 10 If length of information varies, the text field is as long as the longest information.
_____ R 11 The text field includes scroll bars if space in the window is limited.
_____ R 12 The text field includes a decimal point if numeric data are entered in decimal format.
_____ R 13 Field format is consistent with users' expectations and presented in meaningful chunks.
_____ R 14 Strings over 5-7 characters long are entered in smaller chunks, separated by a delimiter.
_____ R 15 Data that are known or can be computed are automatically entered in a field.

Noneditable Text Fields

_____ M 1 Editable text is presented in a text field.
_____ M 2 Labels such as titles, headings, and directions are presented directly in the window.
_____ R 3 Dynamic noneditable text is presented directly in a window or in noneditable text field.
_____ R 4 Each type of text is distinguished in terms of color.
_____ R 5 A noneditable text field has a different appearance than an editable text field.
_____ R 6 When the pointer is in a noneditable text field, its shape does not change to an I-beam.
_____ R 7 Clicking on noneditable text field does not change its appearance or display a text cursor.

Text Entry in a Text Field

_____ R 1 Variable-length text is automatically justified or truncated during text entry.
_____ R 2 Numeric data can be entered from the keyboard or the numeric keypad.

_____ R 3 The amount of data users have to enter in a text field is kept to a minimum.
_____ R 4 Automatic entry of data into a text field is performed whenever possible.
_____ R 5 Windows contain an indicator when position hooking to fill a text field is available.
_____ R 6 Users are not required to enter data in a mandatory field before moving to another field.
_____ R 7 Users are not required to correct an error in data entry before moving to another field.
_____ R 8 Users can enter data in whatever order they choose, with errors flagged when they occur.
_____ R 9 Users cannot commit data until all mandatory data entered & all errors corrected.
_____ R10 When users make an error, they receive feedback that an error has occurred.
_____ R11 Users can fix errors by editing individual characters, not erasing/retyping entire field.
_____ R12 If data being entered are unrelated to one other, users control when error checking occurs.
_____ R13 If data being entered are interdependent, error checking occurs on a field-by-field basis.

Navigation Between and Within Text Fields

_____ R 1 The text cursor moves between areas with the Tab or arrow keys, not automatically.
_____ R 2 When the text cursor moves into a text field, it appears at the left end of the field.
_____ R 3 The size of the text cursor changes to match the size of text in the field.
_____ R 4 When number of entries in text field is small, pop-up menu with entries can be provided.
_____ R 5 Autotabbing used only when data broken into smaller groups, entered in separate fields.

LIST BOXES

Appearance

_____ M 1 The items in a list are displayed vertically, with one item per line.
_____ R 2 If list includes a title, it describes the purpose or contents of the list.
_____ R 3 The title appears above the list box and is not followed by a colon.
_____ M 4 A vertical scroll bar appears to the right of the list when items exceed space available.
_____ R 5 A list scrolls only in response to user action and does not scroll automatically.

_____ R 6 A list is wide enough to read the items without having to scroll horizontally.
_____ R 7 List items appear in sequential order based on nature of items and sequence expected.
_____ R 8 A list box is wide enough for users to read all items without scrolling horizontally.
_____ R 9 If items differ in length, list is wide enough for average items, has horizontal scroll bar.
_____ R10 If default item is designated, it is selected (highlighted) when list is displayed.
_____ R11 The default is first item in list, item likely to be selected, or item(s) previously selected.
_____ R12 Content of list box with view-only items has appearance & behavior of noneditable text.
_____ R13 A list box displays 6-8 items at a time, or all items if there are fewer than 6.

Behavior

_____ R 1 Items added to a list appear in their correct position in the list, not at the end of the list.
_____ R 2 Users can quickly determine where in a list to expect a new item to appear.
_____ M 3 Users search a list by moving the scroll bar slider until the item appears.
_____ R 4 Speed search: Type first letter of item; list scrolls to first instance with that letter.
_____ R 5 Incremental search: Type first few letters and press Return; list scrolls to first match.
_____ R 6 Speed search and incremental search are not case-sensitive.
_____ R 7 If search has to be case-sensitive, then this information is provided to users.
_____ R 8 Feedback is provided when no match is found in a speed search or incremental search.
_____ R 9 If Motif selection box is used, it behaves as in an incremental search.
_____ M10 The S button on the pointing device is used to select item(s) in a list box.
_____ M11 Space (and Select) select item(s) from the keyboard.
_____ M12 If window has default action, double clicking on item chooses item & executes the action.
_____ R13 Selecting an item does not affect the order of the items in the list.

Multi-Column List Boxes

_____ R 1 If multiple items can be selected, label is included to provide this information.
_____ R 2 If multiple items can be selected, feedback is provided on number so far selected.

_____ R 3 Users sort records in a multi-column list box by selecting button in column heading.
_____ R 4 When selected, the button remains highlighted to indicate the column that was sorted.
_____ R 5 Additional sort variations in list matrix are provided in pull-down menus/push buttons.

List-to-List Transfer

_____ R 1 Transfer window has source & destination lists, push buttons to transfer between lists.
_____ R 2 The push buttons contain text labels or arrows indicating the direction of the move.
_____ R 3 Push buttons are available/unavailable based on direction of transfer between lists.
_____ R 4 Users can transfer multiple items, not multiple instances of item to destination list.
_____ R 5 The window can include radio/ check buttons to modify the contents of the source list.
_____ R 6 An item in source list can be copied or moved when transferred to the destination list.
_____ R 7 If transfer is a copy, item is marked when transferred, unmarked when transferred back.

SCROLL BARS

Appearance

_____ M 1 Users are able to scroll to the top or the bottom of the information but not beyond.
_____ M 2 Relative slider position indicates relative position of information displayed in window.

Behavior

_____ M 1 Pressing S button on stepper arrow moves one unit in arrow direction.
_____ M 2 Pressing S button on trough moves one page (less one unit) in direction indicated.
_____ M 3 Dragging the slider with S button moves slider in pointer direction.
_____ M 4 Pressing T button on trough moves slider to pointer position.
_____ M 5 Dragging T button on trough moves slider to pointer position, with pointer movement.
_____ M 6 Cancel returns the slider to its position before the sliding operation began.
_____ R 7 Dragging past top/bottom of scrollable area makes window scroll in pointer direction.

_____ R 8 Speed of autoscrolling is same as when users press on a stepper arrow.
_____ R 9 Autoscrolling stops when pointer moved into or outside window or stop drag action.
_____ M10 Arrow keys move slider one unit in arrow direction.
_____ M11 Ctrl with arrow keys move slider one large increment in arrow direction.
_____ M12 PageUp, PageDown, PageRight, and PageLeft page in specified direction.
_____ M13 Ctrl+Begin and Ctrl+End scroll to beginning/end of scrollable region.

SCALES

Appearance

_____ R 1 A scale contains tick marks and is labeled with minimum/maximum values for the scale.

Behavior

_____ M 1 Pressing S button on stepper arrows moves one unit in arrow direction.
_____ M 2 Pressing S button on scale bar moves one large increment at a time in direction indicated.
_____ M 3 Dragging the slider with S button moves slider in pointer direction.
_____ M 4 Pressing T button on trough moves slider to pointer position.
_____ M 5 Dragging T button on trough moves slider to pointer position, with pointer movement.
_____ M 6 Cancel returns the slider to its position before the sliding operation began.
_____ M 7 Arrow keys move slider one unit in arrow direction.
_____ M 8 Ctrl with arrow keys move slider one large increment in arrow direction.
_____ M 9 Ctrl+Begin and Ctrl+End move slider to minimum/maximum scale values.

Gauges

_____ R 1 A gauge is a scale used to present values that users cannot change.
_____ R 2 A gauge indicates the units of measurement represented by the gauge.
_____ R 3 Trough region fills dynamically to indicate the relative amount of processing completed.
_____ R 4 Gauge includes label with current percent value; trough & label updated dynamically.
_____ R 5 A gauge does not include a slider or arrow buttons.

COMBINATION CONTROLS

Combination Boxes

_____ R 1 A combo box contains an editable single-line text field & a list box below the text field.
_____ R 2 Users select an item from the list to display in the text field or type directly in the field.
_____ R 3 When users select an item from the list, it replaces any text in the field.
_____ R 4 The text entered in the text field does not have to match an item in the list.
_____ R 5 The text entered in the text field is not added to the list.
_____ R 6 The list is large enough to display 6-8 items at a time, or all of the items if fewer than 6.
_____ R 7 A vertical scroll bar is provided when the list is too long to see all of the items.
_____ R 8 The combo box is wide enough for users to read all of the items in the list.
_____ R 9 The text field is the same width as the list.
_____ R 10 List items appear in sequential order based on nature of items and sequence expected.
_____ R 11 When displayed, the text field can either be empty or pre-filled with default list item.
_____ R 12 The default entry is highlighted when text field has focus so typing overwrites entry.

Drop-Down Combination Boxes

_____ R 1 Drop-down combo box has editable field, arrow button, & list box when button depressed.
_____ R 2 Users can view list & select an item to display in text field or type directly in the field.
_____ R 3 A drop-down combo box has same appearance as a combination box.

Spin Buttons

_____ R 1 A spin button allows users to enter no more than 20 discrete, ordered values.
_____ R 2 Spin button has single-line text field & up/down arrow buttons to the right of the field.
_____ R 3 Text field is editable (if all values not included) or noneditable (if all values included).
_____ R 4 When displayed, the text field contains a default value.
_____ R 5 Users click on the up /down arrows to “spin” increase/decrease) the entry in the field.
_____ R 6 When largest/smallest value reached, entries wrap to cycle continuously thru full range.
_____ R 7 If the text field is editable, users can type a value directly in the field.
_____ R 8 If used for DTG or lat/long, separate spin buttons are provided for each part of the entry.
_____ R 9 Spin buttons can be combined with standard text fields for data entry.

STANDARD AND NONSTANDARD CONTROLS

Consistent Appearance and Behavior

_____ R 1 All of the controls in a window are identifiable solely on the basis of their appearance.
_____ R 2 All controls with the same function have the same appearance.
_____ R 3 Controls that are similar in shape have distinctive visual cues.
_____ R 4 Text/graphics in a window are clearly different in appearance from standard controls.

Using Non-Standard Controls

_____ R 1 If nonstandard control is used, it has as much of standard “look and feel” as possible.

Adapting Controls When Using Commercial Software

_____ R 1 COTS software is configured to be compliant with the specifications presented here.
_____ R 2 Changes made to controls in COTS software do not conflict with standard look and feel.

SYSTEM LOGIN

Login Procedure

_____ M 1 Users complete a login procedure before system functions can be accessed.
_____ M 2 A system makes available only the applications to which a user is allowed access.
_____ R 3 If appropriate, users also complete a login for individual or groups of applications.
_____ R 4 If system is unavailable, message appears indicating system status and when available.
_____ M 5 A login window appears on the screen when users begin a session on a system.
_____ M 6 A login window contains two text fields for entering user identification and password.
_____ M 7 The text fields do not provide clues as to the number of characters required.
_____ M 8 The appearance and behavior of objects in login window is consistent with table 7-1.
_____ M 9 Users enter a valid identification and password before a session is initiated.

_____ M10 If invalid identification / password entered, an error message appears in the window.
_____ M11 Users who fail repeatedly to log on are locked out & must contact system administrator.

System Start-Up

_____ R 1 During start-up, “unavailable” message displayed; pointer is a watch; input is disabled.
_____ R 2 When system ready, message disappears, pointer is standard shape, & input is enabled.
_____ R 3 If appropriate, system displays status messages, e.g., response time, unavailability.
_____ M 4 The system window has a default initial appearance following start-up.

Classification Markings

_____ M 1 The classification marking indicates the highest level of data currently displayed.
_____ M 2 Unclassified bar is green, Confidential is blue, Secret is red, Top Secret is orange.
_____ M 3 Classification colors are displayed as background in the classification bar.
_____ M 4 Classification text is in upper-case letters, at least 14-pt bold font.
_____ M 5 Classification terms are spelled out, with caveats abbreviated per security directives.
_____ M 6 There are no embedded spaces within words in the classification label.
_____ R 7 Alternate classification color sets are available in a system.

MENU-BASED INTEGRATION

System Window Design

_____ M 1 The system window is displayed when system start-up is complete.
_____ M 2 The system window covers the entire screen.
_____ M 3 The system window contains classification & menu bars at the top; title is optional.
_____ M 4 The current classification level appears in the middle of the classification bar.
_____ R 5 A digital clock is displayed to right of classification bar, and status (e.g., alerts) to left.
_____ M 6 The system menu bar lists the titles of the menus available at the system level.
_____ M 7 These menus provide access to the application programs available within the system.

_____ R 8 Action icons common to all applications are placed along left margin, from the top down.
_____ R 9 Icons for application windows are displayed at the lower left corner of the screen.
_____ R10 Window icons for each application indicate functionality provided by the application.
_____ R11 The system window implements the color set indicated in table 7-1.
_____ M12 The system window cannot be moved or resized.
_____ M13 The classification and system menu bars cannot be obscured by application windows.
_____ M14 The system window is always active so users can select help / system-level menu options.

Integration of Applications into System-Level Menus

_____ R 1 Applications are integrated into system according to functionality each provides.
_____ R 2 Applications that display geo-related tactical info are integrated into a map window.
_____ R 3 Applications that provide generic functionality are integrated into system menus.
_____ R 4 Application components may distributed among multiple menus based on functionality.
_____ R 5 A system controls whether users can access an application or function in an application.
_____ R 6 System deactivates keyboard accelerators for functions to which users are denied access.
_____ R 7 A system disables application-defined keyboard accelerators in system-level menus.
_____ R 8 System defines order of application functions in system menus, ensures order is followed.
_____ R 9 Cascading submenus are used, whenever possible, to reduce menu length.
_____ R10 Users are able to view and select all of the options in lengthy system menus.

The System Menu Bar

_____ R 1 The menu titles in the system menu describe the overall functionality of the system.
_____ R 2 The menu bar contains no more than ten menu titles plus Help.
_____ R 3 Menu titles begin at the left margin and extend rightward; are spaced to be easy to read.
_____ R 4 Each menu title and menu option includes a mnemonic.
_____ R 5 Keyboard accelerators in system menu do not conflict with those in applications.

_____ R 5 Leftmost menu titles provide functions related to system, chart, and comms functions.
_____ R 6 Middle menu titles provide functions that are specific to the particular system.
_____ R 7 Rightmost menu titles provide support and miscellaneous functions.
_____ R 8 Help appears at the far right margin of the system menu bar.
_____ R 9 A system uses the same menu titles as other systems providing same functionality.
_____ R10 An application has same name whenever it is part of a system, is in same system menu.

Access to System and Application Functions

_____ R 1 A system integrates common application functions so they are accessed in single window.
_____ R 2 System menu options available within an application are displayed in normal text.
_____ R 3 Options that are inappropriate to execute from within the application are dimmed.
_____ R 4 If all of the options in a system menu are inappropriate, the menu title is dimmed.
_____ R 5 If only one instance of an application can run, the menu option is dimmed while running.
_____ R 6 If multiple instances can run, menu option returns to normal to indicate it can be selected.
_____ R 7 Users can select Help from the system menu bar at any time.
_____ R 8 Users can browse all titles in system menu bar even if all menu options are unavailable.

DESKTOP-BASED INTEGRATION (TBD)

SYSTEM SUPPORT

System Support Functions

_____ R 1 A system includes resources and utilities needed to support overall tactical functionality.
_____ R 2 System resources include logout, print, status, preferences, alerts, peripherals, & files.
_____ R 3 System utilities include basic features, COTS software, macros, and briefing support.
_____ R 4 Users can print an entire screen or a window with/without secondary windows.
_____ R 5 Printing includes the classification marking and produces an accurate color shading.

_____ R 6 Users can work with files without detailed knowledge of the file structure.
_____ R 7 A system includes a screen saver and provides a means to rapidly suppress a display.
_____ R 8 Users can temporarily suspend a session without logging out.
_____ R 9 During suspension, processing continues; interaction not allowed until session resumed.
_____ R 10 The system provides an Active Windows option if users work in multiple windows.

User Preference Settings

_____ R 1 A system provides the user-specified settings needed to support overall system functions.
_____ R 2 Users can set parameters only for system functions to which they are granted access.
_____ R 3 A default value for each parameter is set for all applications unless changed by users.
_____ R 4 Users can review the parameters and reset them at any time during a session.
_____ R 5 When users end the session, session-specific settings revert to the default values.

System-Level Navigation Aids

_____ R 1 A system implements navigation aids for quick access to specific functions in the system.
_____ R 2 "Access" aids provide alternate view of system functions, navigate directly to windows.
_____ R 3 A system provides navigation aids tailoring system menus to the task being performed.
_____ R 4 "Tailoring" aids let users hide/show menu options for modules with related functions.
_____ R 5 "Tailoring" aids do not affect the underlying processes for any of the modules.
_____ R 6 Navigation aids can be provided as a system-level menu or take the form of a palette.

SYSTEM LOGOUT

_____ R 1 Users select Logout from the appropriate menu to end a session.
_____ M 2 Users are prompted to confirm Logout if there are any unsaved data.
_____ R 3 Users are prompted to log out of any applications that required they login.
_____ M 4 During logout, all processing in application windows stops, & all windows are closed.
_____ M 5 When logout is complete, the initial login window is displayed.

_____ R 6 If there is auto logout, system has standard length of user inactivity before logout occurs.
_____ R 7 Users can modify the time period before auto logout occurs.
_____ R 8 A message is displayed during inactivity indicating action needed to avoid auto logout.
_____ R 9 An auditory signal is presented at intervals during the period of inactivity.
_____ R 10 For auto logout, unsaved data is saved, with a message indicating logout and file name.

APPLICATION WINDOW DESIGN

_____ M 1 All primary and dialog windows contain a title and a main area.

Window Title

_____ M 1 A window title appears in the title bar of a window.
_____ M 2 The title is centered in the title bar and presented in upper-case letters.
_____ R 3 Each window title in an application is unique.
_____ R 4 If application has multiple primary windows, each title indicates the window purpose.
_____ R 5 The title of a dialog window describes its purpose and includes the application name.
_____ R 6 The title does not contain version / path information, does not present dynamic info.
_____ R 7 The title of a dialog window matches the wording of the menu option that displayed it.

Window Menu Bar

_____ M 1 If a window includes a menu bar, it appears below the title bar.
_____ R 2 The menu bar contains no more than ten menu titles plus Help.
_____ R 3 The titles begin at the left margin and extend rightward, with Help at right margin.
_____ R 4 Commands (e.g., push buttons) are not included in a menu bar.
_____ R 5 Each menu title includes a mnemonic.
_____ R 6 If one of the menus in a window can be torn off, all in the window have this feature.
_____ R 7 Options in application menus are unique and do not duplicate functions in system menu.
_____ R 8 Application menu titles are unique and do not duplicate the titles of system menu.
_____ R 9 A menu has same title and contains same options whenever it is in application menu bar.

_____ R 10 Motif conventions concerning menu design and content are followed to the extent possible.
_____ R 11 Application menu structures conform to the specifications presented in this style guide.
_____ R 12 First menu contains options for users to work with the data in the window as a whole.
_____ R 13 The first menu is File or an application-specific term with comparable meaning.
_____ R 14 The first menu includes an option to exit the application.
_____ R 15 Options in the File menu are: New, Open, Save, Save As, Print, Close, Exit.
_____ R 16 An Edit menu, if included, contains options for users to modify the data in the window.
_____ R 17 Options in the Edit menu are: Undo, Cut, Copy, Paste, Delete, Select All, Deselect All.
_____ R 18 Help menu provides access to additional information about the content of the window.

Arrangement of Window Controls

_____ R 1 Controls performing similar/related functions are presented together in a control panel.
_____ R 2 A control panel has a frame and is clearly labeled to indicate the function it performs.
_____ R 3 If the panel has a title, it is in upper and lower case letters and not followed by a colon.
_____ R 4 The controls are grouped into rows or columns as appropriate to the function.
_____ R 5 Controls for similar options are together; different options are grouped separately.
_____ M 6 If scroll bars are needed, they are located to right or at bottom of area being scrolled.
_____ R 7 Scroll bars scroll main part of window only and not menu bar or message bar in window.
_____ R 8 When a window is displayed, all of the controls reflect current state of the application.
_____ R 9 In modeless window, changes to application are updated in window while displayed.
_____ R 10 A default option can be defined in a control if there is an expected choice in the window.
_____ R 11 Default choice is selected (i.e., highlighted) when the window is initially displayed.
_____ R 12 Every window provides a push button that allows users to dismiss the window.
_____ R 13 When window displayed, check buttons should appear in an unselected state.

_____ R 14 Controls that are temporarily unavailable are dimmed and not available for selection.
_____ R 15 Controls that are never available to users do not appear in a window.

Use of Borders and Frames

_____ R 1 Lines, borders, & frames are used to group related controls, separate from unrelated ones.
_____ R 2 A thin, single-line border surrounds a group of controls.
_____ R 3 No more than 3 line weights are used at one time; lines are consistent in height & length.
_____ R 4 Frames & borders are not overused, do not result in unnecessary clutter within a window.
_____ R 5 The heading or title for the group of controls is placed inside the border or in the border.
_____ R 6 The heading is either left justified or centered within the frame.
_____ R 7 If heading longer than text in controls, frame size is extended to be wider than heading.

Arrangement of Push Buttons

_____ R 1 The push buttons are displayed at bottom of window, in a row that is centered.
_____ R 2 Push buttons are ordered left to right based on sequence of use, most often used on left.
_____ R 3 Buttons for positive actions are on the left, followed by negative and canceling actions.
_____ R 4 If a default push button is included in a window, it is the leftmost button.
_____ R 5 A Help button is included in every window and is the rightmost button.
_____ R 6 Push buttons appear in the same order throughout the application.
_____ R 7 If push button actions affect different objects, their labels reflect what each affects.
_____ R 8 Push buttons are arranged near the object(s) to which they are related.
_____ R 9 Push buttons related to overall window functionality are at the bottom of the window.
_____ R 10 Separate push buttons provided for mutually exclusive actions; unavailable one grayed.
_____ R 11 A window contains push buttons for all of the functionality provided by the window.
_____ R 12 Close and Cancel are not included as push buttons in the same window.
_____ R 13 If an expected choice cannot be anticipated, there is no default defined for the window.
_____ R 14 Push button order in modal windows: OK / Cancel / Help.
_____ R 15 In modeless windows: OK / Apply / Cancel / Help or OK / Apply / Reset / Cancel / Help.

_____ R 16 OK is the default in modal windows and in modeless windows that are transient.
_____ R 17 Apply is the default in modeless windows that are displayed for multiple actions.

Palettes and Icon Bars

_____ R 1 A palette is available when users interact frequently with objects / actions in window.
_____ R 2 Palette provides rectangular radio buttons defining related exclusive modal choices.
_____ R 3 The choices in a palette are presented as icons rather than text whenever possible.
_____ R 4 The choices in a palette are arranged vertically at the left or right edge of a window.
_____ R 5 An icon bar includes buttons for applying settings and executing actions.
_____ R 6 An icon bar is positioned horizontally across top of a window, below the menu bar.
_____ R 7 Palette / icon bar provides redundant access to actions available elsewhere in a window.
_____ R 8 A palette or an icon bar contains no more than 20 buttons of equal size.
_____ R 9 The contents of each palette button are large enough to be read and selected easily.
_____ R 10 The meaning of the icons in a palette is consistent within / between applications.
_____ R 11 Normal appearance of palette button is "flat;" when selected, button is recessed.
_____ R 12 The buttons in a palette are positioned so their edges are nearly touching.
_____ R 13 Buttons are arranged in order expected by users, frequency / sequence of use / importance.
_____ R 14 A palette is movable, supports keyboard navigation, & can be removed from the window.
_____ R 15 An alternative in a palette changes its appearance to indicate its selected state.
_____ R 16 Choices that become unavailable change appearance to show they cannot be selected.
_____ R 17 An alternative remains selected as long as the mode invoked by alternative is in effect.
_____ R 18 The pointer shape changes to indicate type of operation users can perform in the mode.
_____ R 19 The pointer retains its modified shape whenever it is in window where mode is in effect.
_____ R 20 If the pointer is outside the window with mode, it changes to the appropriate shape.

_____ R 21 Palette provides a choice for returning to neutral state or has auto return to neutral state.

Visual Design Considerations

_____ R 1 Window components are sized, spaced, & positioned per recommendations by Kobara.

Message Bar

_____ R 1 A message bar presents noncritical application messages/info on nonstandard actions.
_____ R 2 The left side of the message bar is for routine messages, simple help, and window status.
_____ R 3 The right side of the message bar is used for information about the window.
_____ R 4 Message area contains read-only text; users cannot type or modify any text in this area.
_____ R 5 A message bar is part of the primary window and may also be part of a dialog window.

DIALOG WINDOWS

Prompt Windows

_____ R 1 Application presents prompt window when requesting info to continue processing.
_____ R 2 A prompt window interrupts processing by the application.
_____ R 3 A prompt window includes message stating information needed and text field for typing.
_____ R 4 The title of a prompt window indicates the process/application generating the prompt.
_____ R 5 The message in a prompt window is phrased in language that is meaningful to users.

General Design Guidelines for Message Windows

_____ R 1 A message window is as small as possible, large enough to display information required.
_____ R 2 A message window has a unique appearance and appears at a standard location.

_____ R 3 Auditory feedback accompanies message windows containing critical information.
_____ R 4 Users can set level of auditory feedback or disable it as needed.
_____ R 5 When routine information is generated in window icon, icon changes appearance.
_____ R 6 When critical information is generated in window icon, a message window is displayed.
_____ R 7 A message window include title, symbol for the type, message, & one/more push buttons.
_____ R 8 The title of message window indicates the process/application generating the message.
_____ R 9 The message itself is meaningful to users and requires no documentation or translation.
_____ R10 The text in a message is left justified within the window.
_____ R11 In messages with more than one sentence, important info is placed at start of message.
_____ R12 Message is worded so that the action required appears as a push button in the window.
_____ R13 Error messages focus on the procedure for correcting the error, not the cause of the error.
_____ R14 For a repeated error, the message includes recommendations for preventive actions.

Error Message Windows

_____ R 1 An application informs users in an error window when an error occurs.
_____ R 2 The window has symbol, message, and OK/Help push buttons.
_____ R 3 An error window interrupts processing by the application.

Information Message Windows

_____ R 1 An application conveys noncritical information to users in an information window.
_____ R 2 The window has symbol, message, OK/Help push buttons.
_____ R 3 An information window does not interrupt processing by the application.
_____ R 4 Application does not use timed-information window, then resume processing on its own.

Question Message Windows

_____ R 1 An application requests clarification of previous user response in a question window.
_____ R 2 The window has symbol, message, Yes/No/Help or Yes/No/Cancel/Help push buttons.
_____ R 3 A question window suspends processing by the application.

Warning Message Windows

_____ R 1 An application presents critical information on user actions in a warning window.
_____ R 2 Application displays a warning window when users attempt a destructive action.
_____ R 3 The window has symbol, message, and Yes/No/Help or OK/Cancel/Help push buttons.
_____ R 4 A warning window suspends processing by the application.

Working Message Windows

_____ R 1 An application displays a working window when processing time is over 5 sec.
_____ R 2 A working window is displayed when users may want to cancel operation in progress.
_____ R 3 A working window has a working symbol, message, and OK/Cancel/Stop push buttons.
_____ R 4 During lengthy processing, a working window is updated to indicate status of processing.
_____ R 5 A working window does not interrupt processing by the application.
_____ R 6 Window stays displayed until action is complete or window doing action is minimized.
_____ R 7 When processing is complete, the working window is removed (without user action).
_____ R 8 Users can cancel operation in progress, with confirmation required if unsaved data lost.

Command Windows

_____ R 1 A command window contains a list box and a text field but no push buttons.
_____ R 2 The list includes a vertical scroll bar when command history exceeds visible area in list.
_____ R 3 The text field is wide enough for users to view and read an entire command.
_____ R 4 A horizontal scroll bar is included only if command lines are unusually long.
_____ M 5 A command window is modeless.
_____ R 6 Users can type a command in the text field or select an item from list to display in field.
_____ R 7 Pressing Return executes command and adds it to the bottom of command history list.

_____ R 8 Tab moves the location cursor between the list and the text field.
_____ R 9 The command history list supports the same keyboard navigation as a list box.

Selection Windows

_____ R 1 A selection window has a list box, a text field, & OK/Apply/Cancel/Help push buttons.
_____ R 2 The list provides a vertical scroll bar when number of items exceeds visible area in list.
_____ R 3 Users can type in the text field or select an item from the list to display in the text field.
_____ R 4 If users type in the text field, the list scrolls to that item in the list.
_____ R 5 If text does not match any items in the list, users are prompted to add the item to list.
_____ R 6 When users select OK or press Return, the selection is executed and the window closed.
_____ R 7 A file selection window allows users to find and select a file for processing.
_____ R 8 File selection window has text fields, list boxes & OK/ Filter/Cancel/Help buttons.
_____ R 9 Users select directory by typing in Filter text field or selecting an item in Directories list.
_____ R10 Users select file by typing in Selection text field or by selecting an item in Files list.

COLOR USE IN WINDOWS

Application Windows, Menus, and Controls

_____ M 1 Application windows implement the color set indicated in table 8-1.
_____ R 2 Modifications in basic appearance/color follow guidelines in sections 6.8/8.3.3.
_____ R 3 If application allows changes in color, users can do so via a preference setting.
_____ M 4 Changing to an alternative color set does not alter security classification colors.

Color Coding of Tactical Information

_____ R 1 Color is used in accordance with standards & in ways that match user expectations.
_____ R 2 If color is used to impart tactical meaning, it is used as a redundant code.
_____ R 3 If color coding is used, each color represents one category of displayed data.

_____ R 4 Color is used consistently within an application and between applications in a system.
_____ R 5 When coding threat, cyan=friendly; red=hostile; green=neutral; yellow=unknown.
_____ R 6 When coding status, green=operational; yellow=caution; red=inoperative.
_____ R 7 Red is used to indicate critical or irreversible options; noncritical options are not red.
_____ R 8 Colors for status are the same throughout application and restricted to that function.
_____ R 9 The meaning assigned to color codes is consistent with ISO guidelines.
_____ R 10 When used for alerting, color is assigned only to info to which attention is directed.
_____ R 11 Color are assigned standard meaning for alert criticality; color has only this meaning.
_____ R 12 Alerting is indicated by assigning color to text info or by adding colored icons to text.
_____ R 13 Applications that use color for alerting adhere to coding conventions in this style guide.

Guidelines on Color Selection

_____ R 1 Highly saturated colors, opposing colors, and colors at spectral extremes are not used.
_____ R 2 White text is not displayed on a black background.
_____ R 3 Blue is not used as a text color or for any critical information.
_____ R 4 Color in message windows appears in the symbol and is not used for text.
_____ R 5 The number of colors for alphanumeric display do not exceed 7, only 4 codes at one time.
_____ R 6 The number of colors for graphical displays do not exceed 8-9.
_____ R 7 Slight shade changes in color are not used to show gradation or choice.
_____ R 8 The background color behind text is not used to show a change in system status.
_____ R 9 Change in system status is signaled by changing the color of an object next to the text.
_____ R 10 Color area exceeds 3-inch square for max color sensitivity at normal viewing distance.
_____ R 11 Color images are displayed on achromatic background, achromatic on color background.
_____ R 11 Color images on color backgrounds contrast in brightness & hue.

TEXT IN WINDOWS

Text Font, Size, and Readability

_____ M 1 Text colors indicated in table 8-1 are used in windows.

_____ R 2 Any alternative text color has sufficient contrast with background to be readable.
_____ R 3 Fixed width, sans serif fonts are used to present text in windows.
_____ R 4 A heavier (two-pixel stroke thickness) font is used so text is readable when dithered.
_____ R 5 Proportional-width fonts are not used in text entry areas.
_____ R 6 Different fonts are used to distinguish static from system/application-generated data.
_____ R 7 Applications that do word processing provide a choice of fonts as user-selectable option.
_____ R 8 Text is of sufficient size and thickness to be readable at normal viewing distance.
_____ R 9 Text characters contain a minimum 7 x 9 dot matrix construction.
_____ R10 Text for a briefing presentation is readable at the normal audience viewing distance.
_____ R11 Text for briefing presentation is minimum 10 x 14 dot matrix format, double stroke width.
_____ R11 Font size is user-selectable option when single font does not satisfy these requirements.

Capitalization

_____ R 1 Titles and major headings are presented in upper-case letters.
_____ R 2 All other text uses a combination of upper- and lower-case, following standard rules.
_____ R 3 A consistent approach to capitalization is used in the application.
_____ R 4 All upper-case letters is used only for acronyms and abbreviations and for emphasis.
_____ R 5 Arabic rather than Roman numerals are used when information has to be numbered.

Acronyms and Abbreviations

_____ R 1 Acronyms/abbreviations are used only if shorter than full name and understood by users.
_____ R 2 Abbreviations are the shortest possible length that will ensure uniqueness.
_____ R 3 Abbreviations are meaningful, recognizable, and used consistently.
_____ R 4 Words not commonly abbreviated are not abbreviated.
_____ R 5 Acronyms/abbreviations comply with AR-310-50, MIL-STDs 12D, 411E, 783D.
_____ R 6 New acronyms that are generated following rules contained in MIL-STD-12D.
_____ R 7 A dictionary is available (e.g., in Help) for decoding abbreviations/acronyms.

Static Text in Windows

_____ R 1 Consistent grammatical structure is used for all noneditable text in windows.
_____ R 2 Wording is consistent and uses familiar terms and task-oriented language of users.
_____ R 3 Blocks of text are broken into smaller, meaningful groups.
_____ R 4 If continuous text is being presented, the maximum line length is 40-60 characters.
_____ R 5 Line lengths of less than 26 characters are not used.
_____ R 6 Paragraphs are kept short and separated by at least one blank line.
_____ R 7 Continuous text is phrased in simple sentences, in the affirmative, and in active voice.
_____ R 8 A sequence of events or steps is presented in the order the steps are performed.
_____ R 9 The referent for “it” or “they” in a sentence is easily identified.
_____ R 10 Normal punctuation rules are followed, and contractions and hyphenation are avoided.

Editable Text in Windows

_____ R 1 Editable text is displayed upper/lower case as appropriate to the task being performed.
_____ R 2 Stored text is shown in standard format; text editing is converted into this format.

Formats for Date/Time and Latitude/Longitude Display

_____ R 1 The format for presenting date information is YYMMDD.
_____ R 2 The format for presenting time information is HHMM[SS]Z.
_____ R 3 The format for date/time group is DDHHMMZ MMM YY.
_____ M 4 Latitude/longitude information is displayed in separate fields, with Lat/Long labels.
_____ R 5 The format for latitude information is D{D}H or DD{MM{SS}}H.
_____ R 6 The format for longitude information is D{D{D}}H or DDD{MM{SS}}H.

Wild Card Characters in Text Searches

_____ R 1 Users can enter wild card characters to search for specific text patterns.
_____ R 2 @ searches for the occurrence of a single upper- or lower-case alphabetic character.
_____ R 3 # searches for the occurrence of a single numeric character.
_____ R 4 ? searches for the occurrence of a single alphanumeric character.
_____ R 5 * searches for the occurrence of zero or more alphanumeric characters.

CONSIDERATIONS IN WINDOW DESIGN

Selecting Controls to Match User Actions

_____ R 1 Primary window(s) are used for primary actions and to present frequently used controls.
_____ R 2 Dialog windows are used for ancillary actions and to present less-used controls.
_____ R 3 Radio buttons, option menu, or list box is used when selecting from set of discrete choices.
_____ R 4 A scale is used when users need to select from a continuous range of values.
_____ R 5 Radio buttons, not option menu, are used when users need to see all settings in a group.
_____ R 6 An option menu is used to present a set of choices that users select from infrequently.
_____ R 7 An option menu is used instead of radio buttons when space is limited.
_____ R 8 A list box is used when selecting from a large group of options (more than 12).
_____ R 9 Radio buttons or an option menu is used when the set of options is not likely to change.
_____ R 10 A list box is used when the options may change.
_____ R 11 Check buttons are used when choosing from up to 7 items in a group; a list box otherwise.
_____ R 12 Push buttons used for frequently used selections, when pointing device already moving.
_____ R 13 Pop-up menus used only when have to access functions without moving pointing device.
_____ R 14 Pop-up menus are not the only method available for accessing operations.
_____ R 15 Option menus are used when setting values or choosing from a set of related items.

Arranging Information to Match User Actions

_____ R 1 Window is designed so users manipulate objects in ways that support task performance.
_____ R 2 Objects are arranged so users can move quickly and easily among them.
_____ R 3 Pointer movement/keystrokes needed to perform task are minimized.
_____ R 4 Amount of hand movement between keyboard and pointing device is minimized.
_____ R 5 Window layout supports natural scanning order (from left to right and top to bottom).
_____ R 6 Window layout is logical to users and appropriate to actions executed.
_____ R 7 Blank/Empty space around text and graphic objects supports efficient scanning.

_____ R 8 Screen density in text windows does not exceed 60 percent of available character space.

Arranging Information by Importance

_____ R 1 The most important information and controls are in the upper left part of the window.
_____ R 2 The objects in a window are arranged to accommodate users resizing the window.
_____ R 3 Task-critical information is visually set apart from other information.
_____ R 4 At least 1 space above/below & 2 spaces before/after separates critical information.

Designing Windows to Minimize User Memory Load

_____ R 1 A window contains all of the information relevant to the task.
_____ R 2 Users perform the task called for in a window without referring to external information.
_____ R 3 Users enter all information pertaining to a particular operation in a single window.

Techniques for Coding Information

_____ R 1 Coding techniques used are the ones most appropriate to task being performed by users.
_____ R 2 Coding techniques are applied consistently throughout the application.
_____ R 3 Capitalization is the sole indication of critical information in a window.
_____ R 4 Bolding/ brightening, color coding, etc. are used to focus attention on critical information.
_____ R 5 Color and sound are used for critical messages; a response is required before terminating.
_____ R 6 Color is used only for required functionality; other coding methods used when possible.
_____ R 7 Flash coding is used only to display urgent information for user attention.
_____ R 8 No more than two levels of flash coding are used.
_____ R 9 Flash rate is 3-5 Hz with equal on/off; for 2 levels, the second is 1-2 Hz.
_____ R10 For flash coding of displayed item, a flashing symbol is used, not blinking the item.
_____ R11 Users acknowledge the event causing the flashing and can suppress it if desired.
_____ R12 Reverse video is used sparingly and does not conflict with Motif conventions.
_____ R13 The number of size codes is five or less; users have to interpret relative, not absolute size.

_____ R 14 The number of shape codes is limited to 10-20, and relate to object / operation represented.
_____ R 15 Color & detail added to shapes are minimum needed to identify meaning of shape.
_____ M16 Auditory signals are used to alert to critical conditions or operations.
_____ R 17 For noncritical auditory alarms, a simple action acknowledges / turns off the signal.
_____ R 18 Auditory signals are intermittent in nature and allow sufficient time to respond.
_____ R 19 Auditory signals are distinctive in intensity and pitch and do not exceed four.
_____ R 20 Signal intensity, duration, and location are appropriate to environment and personnel.
_____ R 21 Volume of signal and choice of sounds as feedback are available as preference settings.
_____ R 22 No more than two styles of type or two weights are used at one time.
_____ R 23 Variations in type sizes are minimized & limited to no more than three at any one time.
_____ R 24 Capitalization is used in headings / labels, emphasize words / phrases in continuous text.
_____ R 25 Capitalization is not used as the sole indication of critical information in a window.
_____ R 26 Underlining is used sparingly and does not conflict with hypermedia conventions.

Dynamic Information in Windows

_____ R 1 For dynamically changing information, users are able to control the rate of update.
_____ R 2 Users can freeze an updated display, then resume at the stoppage or the current time.
_____ R 3 When reading dynamically changing information, update values no more than once / sec.
_____ R 4 When identifying rate of change or reading gross values, update values 2-5 times / sec.
_____ R 5 Users are alerted to critical information in an inactive or minimized window.
_____ R 6 Users are prompted to return to auto updating after freezing a dynamic window.
_____ R 7 Users are informed if significant changes in data occurred while the display was frozen.

Consistency in Design Across Windows

_____ R 1 A consistent organizational scheme for key elements is used in all application windows.
_____ R 2 The same window design is employed whenever users perform the same basic task.
_____ R 3 Different or distinctive elements can be used to fit a task, if consistent across windows.

Exiting an Application

_____ R 1 Ability to exit an application is provided only in appropriate windows.
_____ R 2 If primary window has a menu bar, Exit is available as an option in one of the menus.
_____ R 3 If primary window lacks menu bar, Exit is available as push button in the window.
_____ R 4 Opportunities provided for exiting an application make sense from a user's perspective.

Designing to Minimize User Error

_____ R 1 Applications are designed to minimize the opportunity for user error.
_____ R 2 Users are not allowed to execute incorrect operation, then informed of error.
_____ R 3 Only those actions that are relevant are available for user selection.
_____ R 4 Controls that cannot be selected are both grayed out and disabled.
_____ R 5 For each object, the actions available match the function users perform with the object.
_____ R 6 Applications do not include controls that do different functions based on current mode.
_____ R 7 Users are not constrained unnecessarily when working in an application.

Designing for Procedural Efficiency

_____ R 1 Applications minimize the number of actions users have to perform in a window.
_____ R 2 Users perform each major task in a single window with all information relevant to task.
_____ R 3 When users launch an application, a task-related window is displayed.

WINDOW MANAGEMENT CONSIDERATIONS

Initial Window Appearance

_____ R 1 A scrollable window displays the topmost information, shows a complete line of text.

_____ R 2 Button settings are saved / revert to default state as appropriate to window function.
_____ R 3 When button(s) are selected but not executed, the selection is not saved.

Initial Window Size and Placement

_____ R 1 Users are able to resize an application window that covers all of the screen.
_____ R 2 Each window is sized so that all objects in it are visible when window first appears.
_____ R 3 A window is wide enough to read the title and tall enough to read title and menu bar.
_____ R 4 A window is displayed at same location where it was when last closed or minimized.
_____ R 5 Window is placed so important information is at center of users' visual focus.
_____ R 6 Window is placed so important information is not obscured.
_____ R 7 Window is placed so amount of pointer movement to execute an operation is minimized.
_____ R 8 When new window is opened, it is positioned on screen so that it is completely visible.
_____ R 9 New window is offset below & to right of an existing window so its title remains visible.
_____ R 10 A window is placed to the right of the information to which it relates.
_____ R 11 If insufficient space, window is placed to left / below / above info to which it relates.
_____ R 12 If unrelated to other windows currently open, a new window is centered on the screen.

Resizing and Maximizing

_____ R 1 When a window is resized, only the window borders and not the size of objects change.
_____ R 2 When a window is resized, the relative position of objects in the window do not change.
_____ R 3 The contents of the window remain visible during resizing so users can view the effect.
_____ R 4 Horizontal and vertical scroll bars appear as appropriate when a window is resized.
_____ R 5 Resizing smaller is limited so objects are not obscured when window is at minimum size.
_____ R 6 Resizing larger is limited so classification / menu bar not obscured by window at max size.
_____ R 7 Pane size / contents may change automatically during resizing based on functionality.

Order of Window Interaction

_____ R 1 When a dialog window is displayed, users can work in other windows as appropriate.
_____ R 2 When a dialog window is displayed, parent window is open/closes as appropriate.

Processing in Minimized Windows

_____ R 1 Users are kept informed when processing is ongoing in a minimized window.
_____ R 2 Users are informed whenever critical processing events occur in a minimized window.

HELP WINDOWS

Access to User Support Resources

_____ M 1 Online support resources are available at system level and in individual applications.
_____ R 2 Help menu in system menu bar provides access to system-level help & other resources.
_____ R 3 Help-related options are help on system, help on keyboard, and system navigation aids.
_____ R 4 Other resources are online documentation, job planner, "how to," training modules.
_____ R 5 Help on system presents information on system capabilities, including software versions.
_____ R 6 Help on keyboard has info on function keys, mnemonics, & accelerators for platform.
_____ M 7 Applications provide functions in Motif help models according to Motif Style Guide.
_____ M 8 Context-sensitive help is accessed by pressing Help, Shift+Help, or Shift+F1, but not F1.
_____ R 9 Windows have a message area that displays information about the control with focus.

Help Window Design

_____ R 1 A help window includes a title, a main scrollable area, and OK push button at bottom.
_____ R 2 The push button in a help window appears and behaves as the default action.
_____ R 3 When a help window appears, it is wide enough to read the text.

_____ R 4 When a help window appears, it displays information at beginning of help description.
_____ R 5 When a help window appears, it does not obscure what it is describing.
_____ R 6 When displayed, help window does not obscure content about which help was requested.
_____ R 7 Help window is placed to right/left/above/below content for which help was requested.
_____ R 8 Help does not disrupt ongoing processing in window about which help was requested.
_____ R 9 Users can move and resize a help window.
_____ R10 Users can keep the help window on the screen while working in another window.
_____ R11 Users can print any help window by selecting all or marking beginning/end of part.
_____ R12 When done with help, users can easily return to where they were working.
_____ M13 Help is available for every primary, secondary, and dialog window in the application.
_____ M14 Help is provided in a window accessed from a Help push button or Help pull-down menu.
_____ R15 If help window is unavailable, Help push button is dimmed & unavailable for selection.
_____ R16 Applications using action icons can provide access to help via icon instead of menu title.
_____ R17 Application windows with a message area can provide help support there.

Help Window Content

_____ M 1 A help window is titled HELP, followed by window name for which help was requested.
_____ R 2 A help window can be moved and resized and is modeless.
_____ M 3 The window ha read-only text area & OK push button centered at bottom of window.
_____ R 4 The window is large enough to display at least ten lines of text in the display area.
_____ R 5 The window is wide enough to display an entire line of text.
_____ M 6 The window contains a vertical scroll bar if the text exceeds the size of the display area.
_____ R 7 The window includes horizontal scroll bar only if requirements dictate smaller windows.
_____ R 8 The window may include Next and Previous push buttons and an index.
_____ M 9 Help for primary & secondary windows has information on purpose & actions available.
_____ M10 Help for dialog windows provides information on actions available.
_____ R11 Purpose section has a short description of the task(s) that users perform in the window.

_____ R 12 The Available Actions section explains each of the actions executed in the window.
_____ R 13 Help windows may explain procedures for performing the task(s) presented in window.
_____ R 14 Help windows may provide supplementary info on data sources, glossary, and shortcuts.
_____ M15 Help text is presented in mixed case, with capitalization to highlight significant info.
_____ M16 Help text is single-spaced, with a double space between paragraphs.
_____ M17 Sufficient space or a separator is used to visually separate the text in each content area.
_____ R 18 Text is bulleted, steps are numbered& explanations are presented in columns.
_____ R 19 Graphics are included only if essential to understand task in the application window.
_____ R 20 Help information is worded concisely, using consistent structure, phrasing, punctuation.
_____ R 21 When presenting a sequence of steps, the explanation follows the same sequence.
_____ R 22 Help information assumes users have had previous training and only need reminding.
_____ R 23 Users can easily scan help text; steps are numbered and in separate paragraphs.
_____ R 24 Help decodes abbreviations / acronyms and explains behavior of user-selectable objects.
_____ R 25 A help window is removed when the related window is minimized or closed.

DATA ENTRY WINDOWS

Data Entry Window Design

_____ R 1 A data entry window includes a title that describes the contents of the window.
_____ R 2 If multiple pages, each page has same title, current page, and total pages indicator.
_____ R 3 If possible, related data fields appear on the same page.
_____ R 4 The numbering of items is continuous from one page to the next.
_____ R 5 The design of each window is consistent with the task being performed by users.
_____ R 6 Data fields are organized by sequence of use, frequency of use, or importance.
_____ R 7 Data fields are organized with related fields together and unrelated fields separated.

_____ R 8 When users work from a hardcopy form, the window format has an identical format.
_____ R 9 Data fields arranged to minimize hand movement between pointing device / keyboard.
_____ R 10 For tabular data, entry areas are arranged in rows / columns, with each one labeled.
_____ R 11 If multiple pages, the row and column labels remain visible along the page edges.
_____ R 12 Every fifth row of data is separated by a blank line or other delimiter.
_____ R 13 Information is automatically justified; numeric data aligned at the decimal point.
_____ R 14 A data entry window contains the controls needed to support data entry & manipulation.
_____ R 15 If a data entry window has more than one page, the window includes paging controls.
_____ R 16 If users have to make multiple entries, the window includes Clear and Restart controls.
_____ R 17 If push buttons are used in a data entry window, they are separate and at the bottom.
_____ R 18 Users can obtain information about a data field and its contents.
_____ R 19 Data field names are unique; the same names are applied throughout the application.
_____ R 20 Data fields are identified as mandatory or optional; optional fields are labeled as such.
_____ R 21 Data fields are text fields (users type) or option menus (users select).
_____ R 22 Data fields with values that cannot be changed are in an information-only area.
_____ R 23 In related fields, labels & text left justified, or labels right justified & text left justified.
_____ R 25 A conditional field is placed to the right of or below the field to which it relates.
_____ R 25 A conditional field is unavailable or not displayed until related control is selected.

Data Entry and Manipulation

_____ R 1 The same set of actions to enter / manipulate data are used throughout the application.
_____ R 2 When window displayed, text cursor is positioned at beginning of first data field.
_____ R 3 Text cursor is placed with pointing device and moved with keyboard / pointing device.
_____ R 4 Do text entry per sec. 2.3.3 & 6.4.3; move per sec. 6.4.4; make selections per sec. 5.3.

_____ R 5 Users can tab to adjacent fields, and tabbing always moves in the same direction.
_____ R 6 Users can execute an Undo command to revert text to its original content prior to typing.
_____ R 7 Users accept default by tabbing to next field in window; tabbing does not affect default.
_____ R 8 If users modify default, change does not affect value when it next appears in the field.
_____ R 9 Users can back up to any field and change it prior to entering data into the system.

Data Validation and Error Checking

_____ R 1 Application checks data validity, gives visual/auditory message when error detected.
_____ R 2 Syntactic error checking is performed to ensure the data are in the correct format.
_____ R 3 Semantic error checking is performed to ensure meaning/context of data entered is correct.
_____ R 4 Syntactic error checking is performed at the time the user tabs out of the data field.
_____ R 5 Users review and correct syntactic errors when they choose to do so.
_____ R 6 Semantic error checking performed as soon as possible, provides similar error feedback.
_____ R 7 Users are given a list of semantic errors when they attempt to enter data in the system.
_____ R 8 Users can enter what they have typed into the system at any time.
_____ R 9 Data are entered with explicit action, e.g., Enter menu option, Apply/OK push button.
_____ R10 When users accept data, all are saved, regardless of text cursor position in window.
_____ R11 If appropriate, application provides feedback to users to indicate successful data entry.

Data Query

_____ R 1 Data query language reflects the perceived structure of the data.
_____ R 2 The language allows users to specify the data to retrieve, then display & manipulate it.
_____ R 3 Users can modify portions of a query without affecting other parts of the query.
_____ R 4 Users can request data without having to tell the system how to find it.
_____ R 5 Queries use operationally meaningful terminology, not reflect how the data are stored.

_____ R 6 Users can construct simple/complex, predefined queries and save/retrieve/execute.
_____ R 7 The language permits alternate forms of the same query using natural language.
_____ R 8 Users are prompted to confirm the processing of a query if retrieval time will be excessive.

TEXT WINDOWS

Text Window Design

_____ R 1 A text window is wide enough to display an entire line of text without scrolling.
_____ R 2 A text window includes vertical scroll bars, indicator of current location in the document.
_____ R 3 Line length is no more than 40-60 characters; text is left justified but not right justified.
_____ R 4 Users can save, access, retrieve, rename, and print the document.
_____ R 5 Users can specify the portions of the document to be printed and the printer.
_____ R 6 The system acknowledges the print command and provides status of printer and queue.

Text Manipulation

_____ R 1 Users can easily specify the format of a document and the font type, size, and style.
_____ R 2 Automatic line break and word wrap at the right margin are available.
_____ R 3 Automatic pagination (page numbers based on the number entered by users) is available.
_____ R 4 The window is formatted as the printed output, or there is option to see this format.
_____ R 5 A copy of the original document is retained until users confirm that it is to be changed.
_____ R 6 The original document is not modified automatically as users make each editing change.
_____ R 7 Applications provide both search and search/replace capabilities for users.
_____ R 8 Search: Users type text string (not case-sensitive); the first instance is highlighted.
_____ R 9 Search/replace: Users type text string to be searched, change desired (is case-sensitive).

TABULAR DATA WINDOWS

Tabular Data Window Design

_____ R 1 Tabular data windows are used for information display only.
_____ R 2 Single- or multi-columned list boxes are provided for users to select one or more items.
_____ R 3 The window includes a title and vertical/horizontal scroll bars if space is exceeded.
_____ R 4 If the data cover several pages, pages are numbered and include push buttons for paging.
_____ R 5 Column labels are identical for all pages; last line on a page is first line on next page.
_____ R 6 The content of the window does not extend over more than one page horizontally.
_____ R 7 A database query output window includes the query that was executed.
_____ R 8 If info can scroll horizontally, column heading scrolls with its associated column.
_____ R 9 If info can scroll vertically, column heading is outside scrolling area & remains visible.

Tabular Data Window Content

_____ R 1 Each tabular data column has a heading; columns are separated by at least 3 spaces.
_____ R 2 Data groupings are indicated by blank space, separator lines, and/or different intensity.
_____ R 3 Multiple colors are used only if the colors provide additional meaning.
_____ R 4 A blank space is inserted after every third to fifth row and column of tabular data.
_____ M 5 Left-justify alphabetic data; right-justify integers; justify decimal data on decimal pt.
_____ R 6 Long strings of numbers are delimited with spaces, commas, or slashes; no leading zeros.
_____ R 7 If a data record extends beyond a line, additional lines are identified as continuations.
_____ R 8 Tabular data are arranged to show similarities, differences, trends, or relationships.
_____ R 9 Data that are important, require immediate response, and/or are frequent are at the top.
_____ R10 If the data extend across several pages, related data are placed on the same page.
_____ R11 The format for the data display is compatible with the format used for data entry.
_____ R12 The ordering and layout of corresponding fields is consistent and has the same names.

GRAPHIC DISPLAY WINDOWS

Graphic Display Window Design

_____ R 1 The window includes a title and is sized so that the entire graphic display is visible.
_____ R 2 Users do not have to scroll or resize a graphic display window to view its contents.

Line Graphs

_____ R 1 Line graphs are used for trends and for spatial, time-critical, or imprecise information.
_____ R 2 The axes of the graph are clearly labeled, include unit of measurement as appropriate.
_____ R 3 The labels are in upper/lower-case letters and oriented left to right for normal reading.
_____ R 4 Minimum/maximum values are shown on each axis, with up to 9 intermediate markings.
_____ R 5 The starting point of an axis is 0, with the gradations indicated in whole numbers.
_____ R 6 Gradations are standard intervals; intervening gradations consistent with labeled scale.
_____ R 7 Labels are used instead of legends or keys to identify the data.
_____ R 8 The labels are oriented horizontally and located next to the data being referenced.
_____ R 9 Each line or curve on a graph is labeled and coded; critical or abnormal data is coded.
_____ R 10 Grid lines are unobtrusive and do not obscure the data presented in the graph.
_____ R 11 Users can display or suppress grid lines as desired.
_____ R 12 The same coding scheme is used consistently within an application.
_____ R 13 A line graph contains no more than four or five lines/curves, and each is labeled.
_____ R 14 If the lines are tightly packed or overlap each other, each one is coded.
_____ R 15 If data are presented in multiple graphs, the same coding scheme is used in each graph.
_____ R 16 Coding is used to highlight important info, distinguish actual from projected data.
_____ R 17 Multiple trend lines are presented on a single graph.
_____ R 18 Users can redraw multiple graphs using the same scale to facilitate comparison.

_____ R 19 For precise values, users can display the actual values on the graph and to zoom.
_____ R 20 When reading exact values is required, application provides aids to interpret scale.
_____ R 21 Lines in a surface chart are stacked above one another to indicate aggregated amounts.
_____ R 22 Area under each line in a surface chart is coded, identified by label displayed in area.
_____ R 23 If a surface chart is used, the data categories are ordered logically.
_____ R 24 If no a priori order, least variable data categories are on bottom, most variable on top.

Bar Charts and Histograms

_____ R 1 Bar charts are used to show several intervals, histograms a large number of intervals.
_____ R 2 Bar graphs have a consistent orientation; comparable bars are adjacent.
_____ R 3 Frequency counts are displayed in vertical bars, time durations in horizontal bars.
_____ R 4 When displayed data are compared with a critical value, a reference index is provided.
_____ R 5 A bar graph is used when number of bars is small; a histogram when number is large.
_____ R 6 Bars in a bar graph are separated, using one-half or less of bar width as spacing.
_____ R 7 Coding is used to distinguish among groups of bars, highlight important data in bars.
_____ R 8 In multiple bar graphs, related groups of bars are presented in consistent order.
_____ R 9 Each bar is identified with its own text label, rather than using a separate legend.
_____ R 10 Stacked bars are used when total measures, portions represented by segments of interest.
_____ R 11 If stacked bars are used, the data categories are presented in the same sequence.
_____ R 12 Data categories are ordered with least variable at bottom of bar, most variable at top.
_____ R 13 Areas within each bar are coded & identified by a text label displayed in the area.
_____ R 14 The design of a bar chart or histogram conforms to user expectations.
_____ R 15 Icons are not used to represent quantitative information.
_____ R 16 Charts and axes are clearly labeled, and important information is highlighted.
_____ R 17 When bars are presented in pairs, they are labeled as a unit and include a legend.

Flow Charts

_____ R 1 Flow charts are used for a schematic representation of sequences or processes.
_____ R 2 The steps presented in a flow chart are ordered logically, by importance, or by certainty.
_____ R 3 If no order identified, the flow chart is organized to minimize length of path through it.
_____ R 4 The shapes (e.g., boxes) used in the flow chart follow existing shape coding conventions.
_____ R 5 If there is no inherent logic, the steps are ordered to minimize the size of the flow chart.
_____ R 6 The path indicated in the flow chart is left to right, top to bottom, or clockwise.
_____ R 7 Each decision point in the flow chart contains a single, simple decision.
_____ R 8 The flow chart elements and lines are consistently coded to assist in understanding.
_____ R 9 The flow chart provides directional indicators to indicate the sequence to be followed.
_____ R 0 A legend describes each element & code; critical information and/or steps highlighted.
_____ R11 The text presented in the chart is oriented for normal reading.
_____ R12 Important elements (e.g., paths) are emphasized through coding.

Pie Charts

_____ R 1 Pie chart is used to provide an approximation of how an entity is apportioned into parts.
_____ R 2 Circle in chart is divided into 5 segments or less; each takes at least 5 percent of circle.
_____ R 3 Each segment is coded in different colors/shadings/textures & identified by text label.
_____ R 4 If segment is too small, the label is placed outside, with a line from it to the segment.
_____ R 5 The label describes content of segment, includes number being represented by segment.
_____ R 6 Segments can be emphasized by special shading, displacing them from rest of chart.

Manipulation of Graphical Data

_____ R 1 Users can select/edit object attributes, change sizes, fill areas with colors/patterns.

_____ R 2 Applications automatically align, complete figures, draw between user-specified points.
_____ R 3 Object selection/transfer methods are available, implemented according to style guide.
_____ R 4 A palette of drawing tools is included as part of the graphics display window.
_____ R 5 Users can draw objects such as lines, rectangles, ovals, and arcs.
_____ R 6 Objects emerge as they are drawn and be easy to reposition, duplicate, and delete.
_____ R 7 Users can group separate objects into a single object.
_____ R 8 Objects are displayed as they will be printed, or there is an option to see this format.
_____ R 9 Original graphics are retained until users confirm that the objects are to be changed.
_____ R 10 The objects are not modified automatically as users change them.

MAP WINDOWS

Map Window Design

_____ R 1 Map controls appear in the map window or are available in separate dialog windows.
_____ R 2 A map window includes identifying information about map and status information.
_____ R 3 Identifying/status information appears in window area or along bottom window margin.
_____ R 4 A continuous indicator of pointer location on map is available in standard window area.
_____ R 5 Maps are displayed using the same orientation, and the important features are labeled.
_____ R 6 Labels are positioned consistently with respect to features, not obscure, & remain legible.
_____ R 7 When displaying color overlays, a color coding key is also displayed.
_____ R 8 Users can display the coding key as desired without having to redisplay the overlay.
_____ R 9 A dialog window displaying a coding key is minimum size and obscures little of overlay.
_____ R 10 If appropriate, coding key is a scale so users can interpret the coding in the overlay.
_____ R 11 Standard symbology and color codes are employed, with help available to users.
_____ R 12 New symbology uses shape coding that is consistent with standard symbology.
_____ R 13 Symbols are placed on the map accurately or connected to the map with graphics.

_____ R 14 If multiple symbol sets are available, users can select a set without losing data.
_____ R 15 Symbols are placed accurately or connected to location with arrows, lines, or graphics.
_____ R 16 Symbol labels appear next to the symbol and present essential information about it.
_____ R 17 The background of the symbol and label are transparent to not obscure other information.

Map Manipulation

_____ R 1 Users can customize a map window to fit the task being performed.
_____ R 2 Users can pan and zoom a map as desired.
_____ R 3 Position or change indicators are provided to return quickly to the normal/starting map.
_____ R 4 When a map is zoomed, size of symbols, labels, and features is adjusted to be readable.
_____ R 5 Users can define a baseline position on a map and return to this position quickly.
_____ R 6 Users can place window, define appearance, select objects, alter tactical appearance.
_____ R 7 Users can add, edit, reposition, delete, determine distance/bearing between points.
_____ R 8 Users can access other functions (e.g., areal computation/verification).
_____ R 9 Latitude and longitude are enterable to the level of accuracy needed.
_____ R 10 Calculations (e.g., range, bearing, position) reflect accuracy appropriate to the scale.
_____ R 11 Graphics drawn on a map include visual indications defining parts that can be selected.

Symbology Manipulation

_____ M 1 Click the S button on a track to select it; other selected tracks remain selected.
_____ M 2 Click S button on multiple tracks one at a time to select them; highlight as selected.
_____ M 3 Drag the pointer over contiguous tracks in a range with the S button to select them.
_____ M 4 Deselect one track by pressing Shift and clicking the S button.
_____ M 5 Deselect all tracks by placing pointer in empty map area & double clicking the S button.
_____ R 6 Users can view or declutter, and obtain additional information for selected symbols.
_____ R 7 The intensity of the map is adjustable to fade out without losing all map features.

_____ R 8 Users can distinguish coincident point symbols and to obtain ambiguity information.
_____ R 9 Users can select categories for automatic update and frequency and rate of update.
_____ R 10 If appropriate, users can temporarily stop and then resume further updates.

MESSAGE HANDLING WINDOWS

Message Preparation

_____ R 1 Message preparation windows follow the same design as data entry windows.
_____ R 2 Users are given basic message header fields, supported in specifying message address.
_____ R 3 Option menus are available for selecting from limited sets of frequently used terms.
_____ R 4 When replying to a message, the appropriate addressee(s) are provided automatically.
_____ R 5 Users can build / maintain lists of common addresses and select from these lists.
_____ R 6 Addresses are checked prior to transmission; users can correct errors before sending.
_____ R 7 Preformatted standard forms are available; format control during entry is automatic.
_____ R 8 Users can specify data, incorporate data files, save during preparation / completion.

Message Transmission

_____ R 1 Message transmission procedures are designed to minimize the user actions required.
_____ R 2 Users can initiate message transmission directly (e.g., select a Transmit push button).
_____ R 3 If a message cannot be sent immediately, it is queued automatically.
_____ R 4 Users can assign message priorities and cancel or abort a transmission.
_____ R 5 Status feedback is available that confirms messages sent & indicates failures.
_____ R 6 Users can specify feedback wanted; an automatic log is maintained.
_____ R 7 Undelivered messages are saved in the event of transmission failure.

Message Receipt

_____ R 1 Users are informed when high priority messages are received.

_____ R 2 Message notification does not interfere, but provides some indication of urgency.
_____ R 3 Incoming messages are automatically queued by time and priority; logs are maintained.
_____ R 4 Users can review summary information, display messages, save/file, and discard.
_____ R 5 A message is displayed in a text window; users can scroll, save, and print it.
_____ R 6 During login, a system provides users with a list of new messages received.
_____ R 7 During session, a system activates an alert indicator to inform users of priority messages.

IMAGERY WINDOWS

Window Design

_____ R 1 An imagery window has an imagery display area & tools for manipulating the image.
_____ R 2 Imagery tools are presented as palette or sets of controls grouped by operation performed.
_____ R 3 Window has identifying information about the image, in standard location in window.
_____ R 4 The window provides scroll bars when entire image does not fit in the window.
_____ R 5 Users can access and retrieve an image for display from a directory of images.
_____ R 6 Users can specify image name or search directory on user-defined criteria and wildcards.
_____ R 7 Users are able to print the image currently displayed in the window.

Basic Viewing Functions

_____ R 1 Users can roam the image in both horizontal and vertical directions.
_____ R 2 Auto/jump/manual/patterned roam control available; users specify rate, direction, area.
_____ R 3 As users roam, they can tag areas for later recall while image remains in display area.
_____ R 4 Window provides default/auto north rotation, interactive rotation in user-defined steps.
_____ R 5 Users can zoom an image, either in predefined steps or in increments they specify.
_____ R 6 Users can select and designate regions of an image for storage.
_____ R 7 Users can create & manipulate geo referenced annotations to display with image.

_____ R 8 Annotations include text, lines, icons, geometric shapes, colors, and patterns.
_____ R 9 Users can manipulate annotation without altering underlying imagery data.
_____ R 10 When image manipulated, scale & orientation of annotations adjust to reflect changes.

Advanced Viewing Functions

_____ R 1 Window has image enhancement filtering capabilities; users can modify bias & gain.
_____ R 2 Users can select/examine a region of interest in an image, without affecting rest of image.
_____ R 3 Users can apply the region of interest to the entire image.
_____ R 4 The window provides mensuration functions and performs isotropic pixel correction.
_____ R 6 Users can create mirror view of image & select alternate vantage points to view image.
_____ R 7 Users can plot geographic data on the image, animate data in forward/reverse direction.
_____ R 8 Users can display a geo outline of geo-referenced imagery on a map or other image.
_____ R 9 Users can register geo-referenced images from same/different sensors, display together.
_____ R 10 Windows with two images provides full geo-referenced image display functionality.
_____ R 11 Users can dissolve/toggle between 2 images, subtract image intensities & display results.
_____ R 12 Users can manipulate separate windows with overlapping geographic coverage.
_____ R 13 User can slave together geo-referenced windows, manipulate concurrently & relatively.

BRIEFING SUPPORT WINDOWS

Window Design

_____ R 1 The window has display area to view/edit slides, tools to manipulate objects on a slide.
_____ R 2 Window includes an icon bar to access commonly used commands for slide manipulation.

_____ R 3 The window is titled with briefing name, has standard location for current slide number.
_____ R 4 The window provides scroll bars when entire slide does not fit in the window.

Building Briefing Slides

_____ R 1 Window provides graphic/text manipulation capabilities as defined in this style guide.
_____ R 2 Graphic features include ability to edit/move/group objects, change attributes/order.
_____ R 3 Word processing features include fonts/type sizes, spellchecking, search/replace.
_____ R 4 Users can select colors for background, lines & text, shadows, title text, fills, & accents.
_____ R 5 Users an specify global features for overall briefing, modify them for individual slides.
_____ R 6 Users can import text/graphics, clip art/animation/video, do/annotate screen captures.
_____ R 7 Users can create a file containing each set of slides (i.e., briefing).
_____ R 8 Content of file includes individual slides in briefing & order for displaying them.
_____ R 9 Users can reorder, add, and delete slides and then save the result.
_____ R10 Users can select & copy multiple slides to re-use in briefing or paste into another briefing.
_____ R11 Users can access/retrieve/rename their briefings, print some/all slides in a briefing.

Presentation Management

_____ R 1 The window includes an option for presenting a slide show.
_____ R 2 Users can select all/some slides, specify slide advance, sequence, timing, transitions.
_____ R 3 All slides have the same output size, aspect ratio, and orientation when presented.
_____ R 4 Keyboard/pointing device can move between slides, stop/restart/end show, blank screen.
_____ R 5 Users can move pointer with pointing device and use it to point to and draw on slides.

GRAPHICAL SCHEDULING WINDOWS

Window Design

_____ R 1 A graphical scheduling window is used to display timelines or scheduled events.
_____ R 2 The schedule has time on horizontal axis, tasks to be performed arrayed vertically.
_____ R 3 Window is titled with schedule name, has scroll bars if schedule does not fit in window.

Window Content

_____ R 1 Events are represented by event icon whose length shows time needed to complete a task.
_____ R 2 The icon is displayed to the right of its associated task.
_____ R 3 Types of events are indicated by color / shading, designator displayed on / above icon.
_____ R 4 Users can access a legend or key that describes the coding technique used in the schedule.
_____ R 5 No more than nine uniquely coded event icons are presented on a schedule at one time.
_____ R 6 Each icon is labeled if more than one event icon is used per task.
_____ R 7 Event icon labels are placed along the vertical axis or on or above the timeline.
_____ R 8 Different scheduling attributes are represented by displaying symbols with event icons.
_____ R 9 Symbols are formed from geometric shapes, fill patterns to show different situations.
_____ R 10 Gridlines are available if schedule is cluttered or users require high degree of precision.
_____ R 11 Gridline is displayed to indicate present date / time; users can show / hide line as needed.

Schedule Manipulation

_____ R 1 Users can define the start / stop time of the schedule to the desired degree of precision.
_____ R 2 Users can display all / part of the preselected duration time.
_____ R 3 Users can select an individual event icon & obtain additional info about the event.
_____ R 4 The selection methods available to users are implemented according to this style guide.
_____ R 5 Users can reschedule an event icon using object transfer methods described in style guide.
_____ R 6 If exact positioning is difficult, users have alternative methods to locate the icon.

APPENDIX C
ACRONYMS AND ABBREVIATIONS

API	Application Programming Interface
ATCCS	Army Tactical Command and Control System
C^{4I}	Command, Control, Communications, Computer, and Intelligence
CIE	International Commission on Illumination
COE	Common Operating Environment
COTS	Commercial Off-the-Shelf
CUA	Common User Access
DIA	Defense Intelligence Agency
DoD	Department of Defense
DODIIS	Department of Defense Intelligence Information System
DTG	Date / Time Group
FM	Field Manual
FOTC	Force Over-the-Horizon Targeting Coordinator
GCCS	Global Command and Control System
GOTS	Government Off-the-Shelf
GUI	Graphical User Interface
HCI	Human Computer Interface
IDHS	Intelligence Data Handling System
IEEE	Institute of Electrical and Electronic Engineers
ISO	International Organization for Standardization
JMCIS	Joint Maritime Command Information System
JTF	Joint Task Force
MIL-HDBK	Military Handbook
MIL-STD	Military Standard
NATO	North Atlantic Treaty Organization
NISE	Naval Integrated System for Exploitation
OSF	Open Software Foundation
POSIX	Portable Operating System Interface for Computer Environments

RGB	Red, Green, Blue
SSC	Standard Systems Center
TAFIM	Technical Architecture Framework for Information Management
TBM	Theater Battle Management
TDA	Tactical Decision Aid
UIC	Unit Identification Code
UIDL	User Interface Definition Language
UIMS	User Interface Management System

APPENDIX D
MAPPING OF VIRTUAL KEYS TO KEYBOARD COMMANDS

Virtual Key	Key Only	Alt+Key	Ctrl+Key	Shift+Key	Alt+Ctrl+Key	Alt+Shift+ Key[65]	C
C			Copy current selection to clipboard		Copy selection to cursor position		
V			Paste clipboard contents				
X			Cut current selection to clipboard		Move selection to cursor position		
Z			Reverse previous operation				
/			Select all elements in a collection				
\			Deselect all elements in a collection				
Backspace	Delete previous character; delete current selection	Reverse previous operation				Re-do previous operation	
Cancel	Stop current operation						
Copy	Copy current selection to clipboard	Copy selection to cursor position					
Cut	Cut current	Move selection					

65. Shaded area (i.e., Alt+Shift+alphanumeric key) indicates key combinations reserved for the execution of macro commands.

	selection to clipboard	to cursor position					

Delete	Delete next character; delete current selection			Cut current selection to clipboard		Move selection to cursor position	
Down	Navigate down one line; navigate down within a tab group		Navigate down one paragraph	Extend selection down one line			Exte sele one
End	Navigate to end of line		Navigate to end of data	Extend selection to end of line			Exte sele of te
Enter	Select menu option; execute default action						
Escape	Stop current operation			Display Window Menu			
F1				Display context-sensitive help			
F2							
F3		Lower a window					
F4		Close a window					
F5	Refresh window content with current data	Restore a window					
F6		Navigate to next window in a family				Navigate to previous window in a family	
F7		Move a window					
F8		Resize a window		Toggle between normal & add mode			
F9		Minimize a window					
F10	Navigate to/from menu bar	Maximize a window		Display/dismiss pop-up menu			
Help	Display context-sensitive help			Display context-sensitive help			
Home	Navigate to beginning of line		Navigate to beginning of data	Extend selection to beginning of line			Exte sele begi text
Insert	Toggle between	Reselect all	Copy current	Paste clipboard	Copy selection		

	insert & replace mode	elements previously selected	selection to clipboard	contents	to cursor location		
Left	Navigate left one character; navigate left in tab group		Navigate left one word	Extend selection left one character			Exte sele one
Menu	Display/dismis s pop-up menu			Navigate to/from menu bar			
PageDown	Navigate down one page		Navigate right one page	Extend selection down one page			Exte sele one
PageLeft	Navigate left one page			Extend selection left one page			
PageRight	Navigate right one page			Extend selection right one page			
PageUp	Navigate up one page		Navigate left one page	Extend selection up one page			Exte sele one
Paste	Paste clipboard contents						
Return	Select menu option; execute default action (except in multi-line text); insert one line in multi-line text		Select menu option; execute default action				
Right	Navigate right one character; navigate right in a tab group		Navigate right one word	Extend selection right one character			Exte sele one
Select	Select element at cursor position			Extend selection to cursor position (except in text)			
Space	Select element at cursor position (except in text); insert a space in text	Activate Window Menu	Select element at cursor position	Extend selection to cursor position (except in text)			Exte sele curs
Tab	Navigate to next tab group (except in multi-line text); navigate to next tab stop in multi-line text	Navigate to next window family	Navigate to next tab group	Navigate to previous tab group (except in multi-line text)		Navigate to previous window family	Nav prev grou
Undo	Reverse						

	previous operation						
Up	Navigate up one line; navigate up within a tab group		Navigate up one paragraph	Extend selection up one line			Exte sele one

INDEX

Flashing (in information coding)	8.5.5.3	8-24	
Flow charts	9.5.1.3	9-16	
Font	8.4.1	8-17	
Frames	8.1.3.2	8-3	
Gauges	6.7.3	6-23	
Global Command and Control System (GCCS)		1.1	1-1
	1.3.2	1-2	
	1.3.3	1-2	
	1.6	1-4	
Graphic display windows	9.5	9-12	
Graphical data manipulation	9.5.2	9-17	
Graphical scheduling windows	9.10	9-24	
Hardware configuration	1.3.1	1-2	
Help windows	9.1	9-1	
Histograms	9.5.1.2	9-14	
Icon bars	8.1.3.4	8-4	
Icons, action	6.1.1.3	6-9	
	8.1.3.4	8-4	
Icons, window	4.6	4-9	
Imagery viewing functions	9.8.2	9-22	
Imagery windows	9.8	9-22	
Immediate feedback	3.5.3	3-13	
Information coding	8.5.5	8-23	
Information message windows	8.2.2.3	8-8	
Input devices	2.0	2-1	
Input devices, alternative	2.4	2-17	
Input devices, interchangeability	2.1	2-1	
Input focus policy	3.1.1	3-1	
Interactive control	3.5	3-13	
Key assignments	2.3.2	2-7	
Keyboard accelerators	5.6	5-13	
Keyboard input	2.3	2-4	
Keyboard operations	2.3.2.1	2-7	
Latitude/longitude format	8.4.6	8-19	
Line graphs	9.5.1.1	9-12	
List boxes	6.5	6-15	
List-to-list transfer	6.5.4	6-19	
Location cursor	2.3.1	2-4	
	3.2.2.4	3-3	
Login procedure	7.1.1	7-1	